Neural Networks in Structured Prediction

November 19, 2015

Last Time

- We talked about using non-structured neural networks to solve structured problems
  - Intuition: neural nets are powerful learners- maybe we don’t need to model statistical dependencies among output variables?
  - Some support for this: POS tagging results…

Goals for Today

- Neural networks in structured prediction:
  - Option 1: locally nonlinear factors in globally linear models
  - Option 2: operation sequence models
  - Option 3: global, nonlinear structured models [speculative]

Locally Nonlinear Models

$$\begin{aligned} score(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(y_{i-1}, y_i, \boldsymbol{x}) \\ &= \mathbf{w}^\top \sum_{i=1}^{n} \mathbf{f}(y_{i-1}, y_i, \boldsymbol{x}) \end{aligned}$$

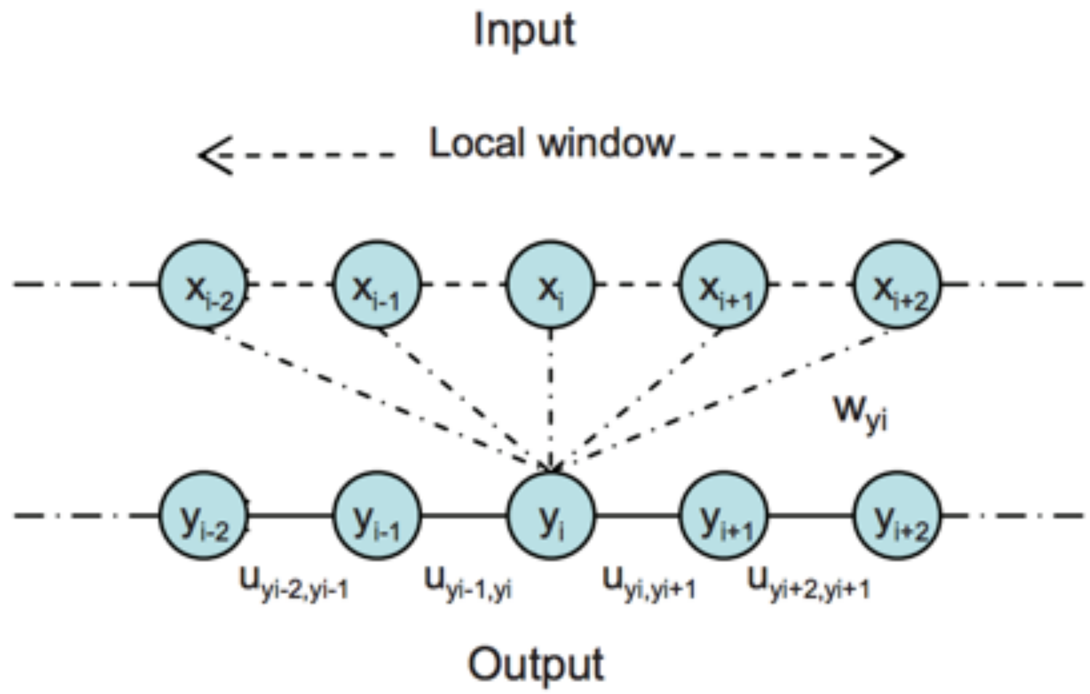
Locally Nonlinear Models

$$
\begin{aligned}
score(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{f}(y_{i-1}, y_i, \boldsymbol{x}) \\
&= \mathbf{w}^\top \sum_{i=1}^{n} \cancel{\mathbf{f}(y_{i-1}, y_i, \boldsymbol{x})} \\
&= \mathbf{w}^\top \sum_{i=1}^{n} \mathrm{NN}(y_{i-1}, y_i, \boldsymbol{x})
\end{aligned}
$$

Local Nonlinear Model

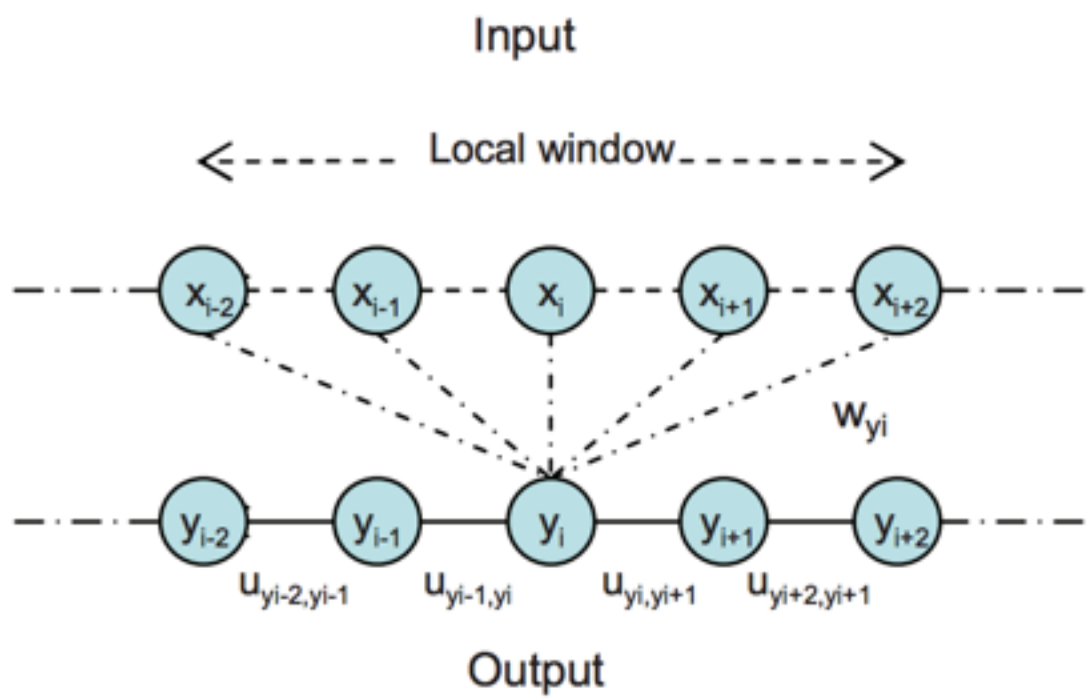
- Neural net returns a vector (a feature vector!) for each local factor
- We still get fast, global decoding using standard linear models
- Feature induction operates locally
- Best of both worlds?

CRF

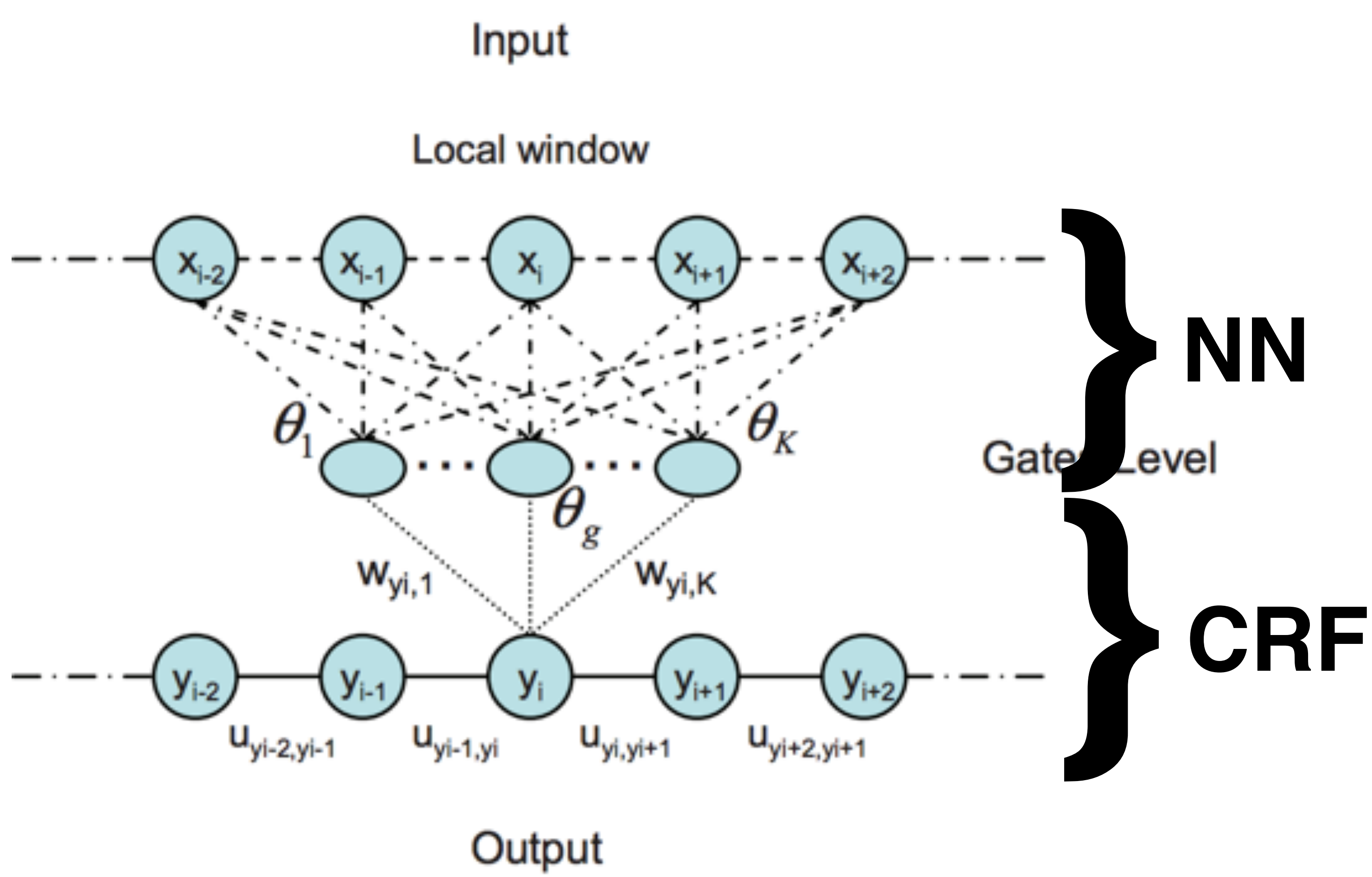


Peng, Bo, Xu (NIPS 2009)

CRF



CNF



Peng, Bo, Xu (NIPS 2009)

Protein secondary structure prediction (Peng et al., 2009)

Methods	Q3(%)
Conditional Random Fields	72.9
SVM-struct (Linear Kernel)	73.1
Neural Networks (one hidden layer)	72
Neural Networks (two hidden layer)	74
Semimarkov HMM	72.8
SVMpro	73.5
SVMpsi	76.6
PSIPRED	76
YASSPP	77.8
SPINE*	76.8
Conditional Neural Fields	**80.1** ± 0.3
Conditional Neural Fields*	**80.5** ± 0.3

Protein secondary structure prediction (Peng et al., 2009)

Methods	Q3(%)
Conditional Random Fields	72.9
SVM-struct (Linear Kernel)	73.1
Neural Networks (one hidden layer)	72
Neural Networks (two hidden layer)	74
Semimarkov HMM	72.8
SVMpro	73.5
SVMpsi	76.6
PSIPRED	76
YASSPP	77.8
SPINE*	76.8
Conditional Neural Fields	**80.1** ± 0.3
Conditional Neural Fields*	**80.5** ± 0.3

Constituency parsing (Durrett & Klein, 2015)

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Dev, all lengths										
Hall et al. (2014)	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30
This work*	**80.68**	**84.37**	**80.65**	**85.25**	**89.37**	**89.46**	**82.35**	**92.10**	**77.93**	**84.68**
Test, all lengths										
Berkeley	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.18	78.53
Berkeley-Tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	80.89
Crabbé and Seddah (2014)	77.66	85.35	79.68	77.15	86.19	87.51	79.35	91.60	82.72	83.02
Hall et al. (2014)	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.17
This work*	**80.24**	**85.41**	**81.25**	**80.95**	**88.61**	**90.66**	**82.23**	**92.97**	**83.45**	**85.08**

Operation Sequence Models

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

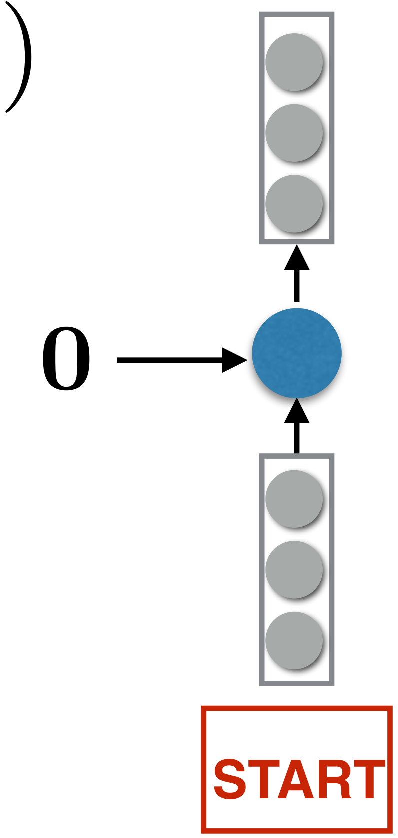
$\mathbf{0} \longrightarrow$

What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \textsc{stop}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

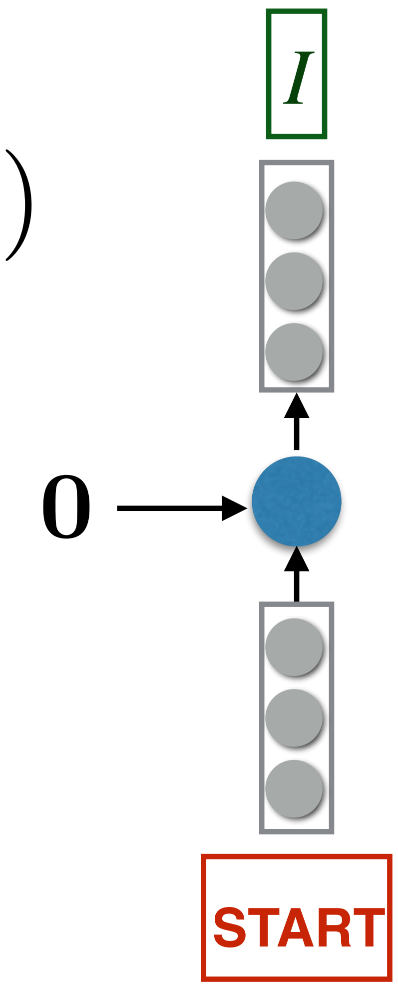


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

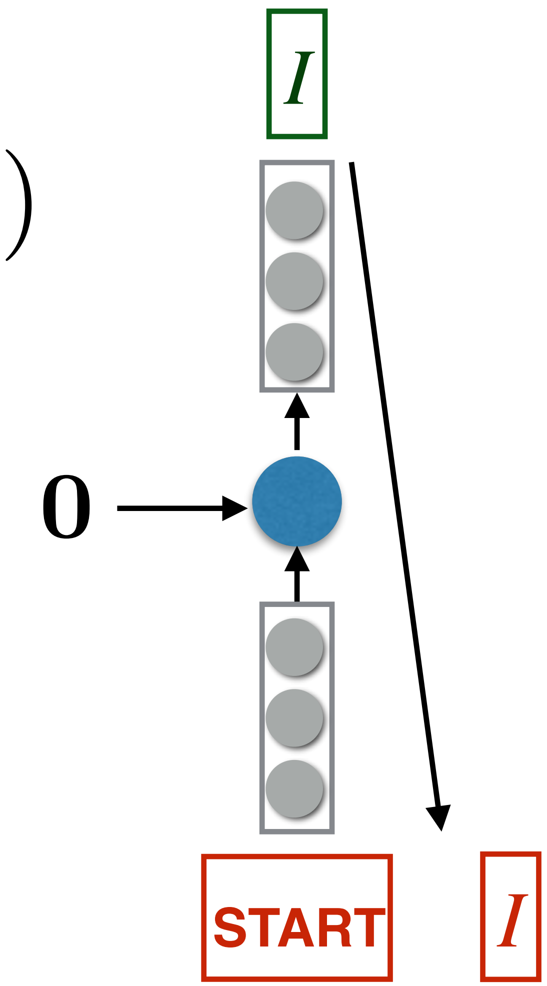


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

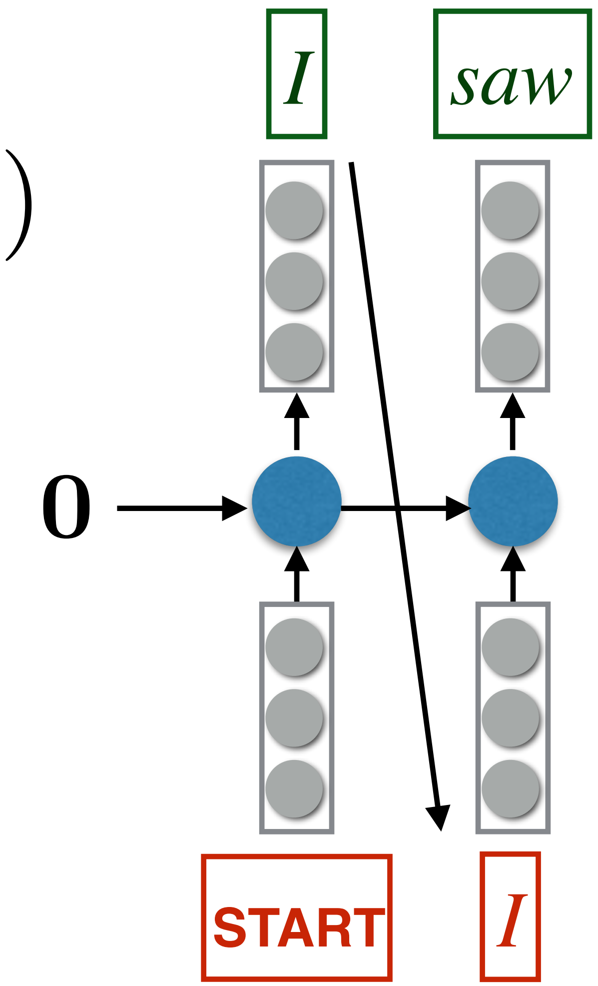


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$
\begin{aligned}
&\text{while } y_t \neq \textsc{stop} \\
&\qquad \mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \\
&\qquad y_t \sim g(\mathbf{h}_t) \\
&\qquad t \leftarrow t + 1
\end{aligned}
$$

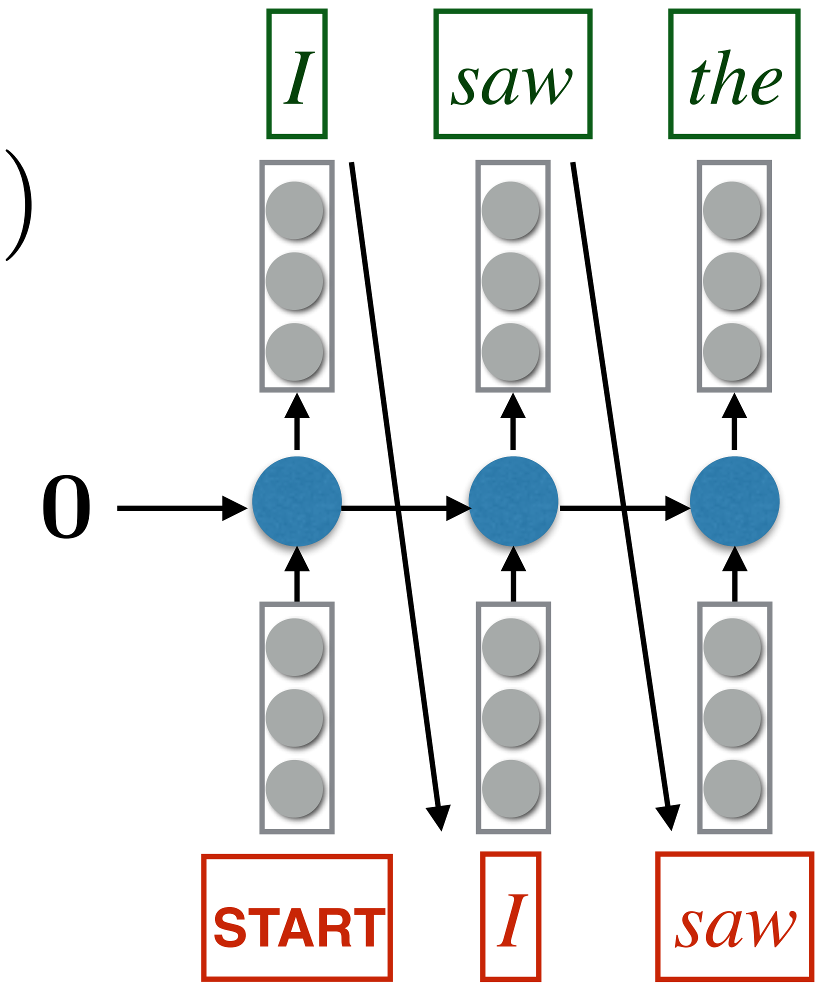


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

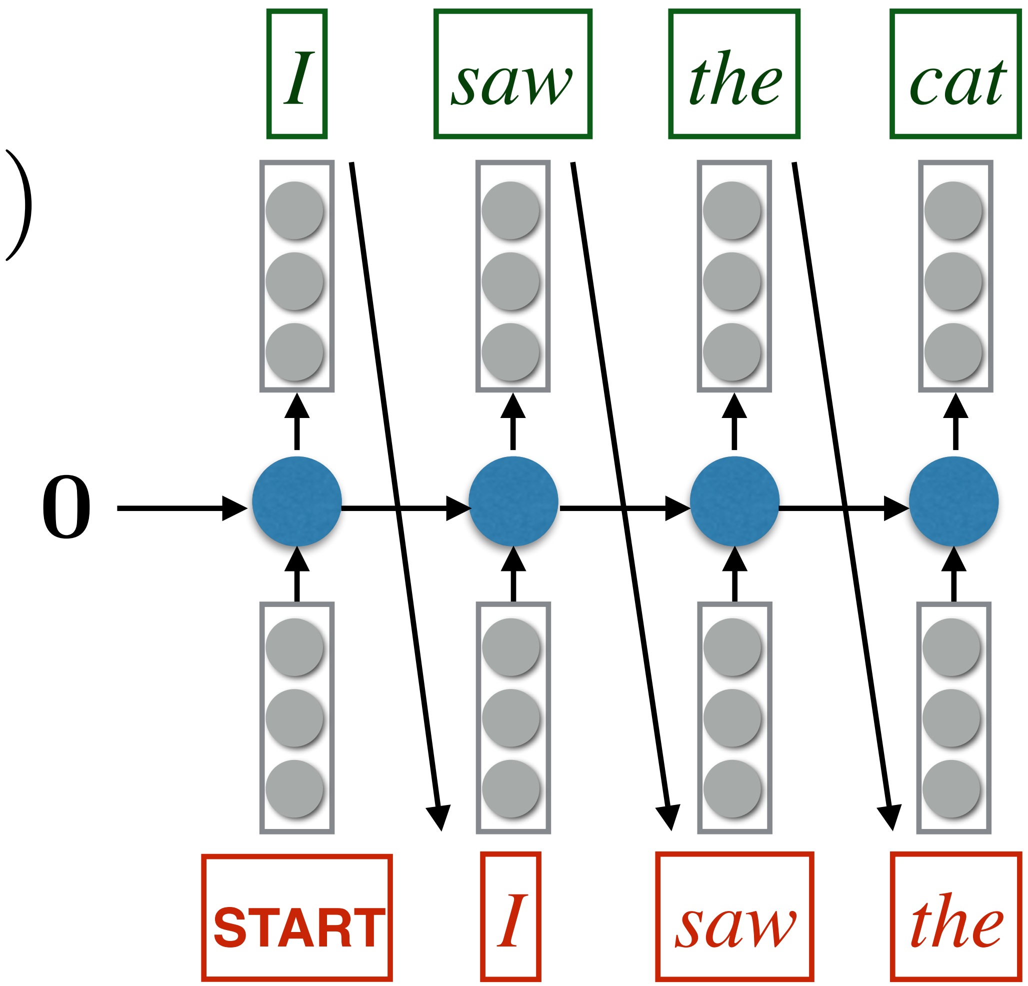


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$

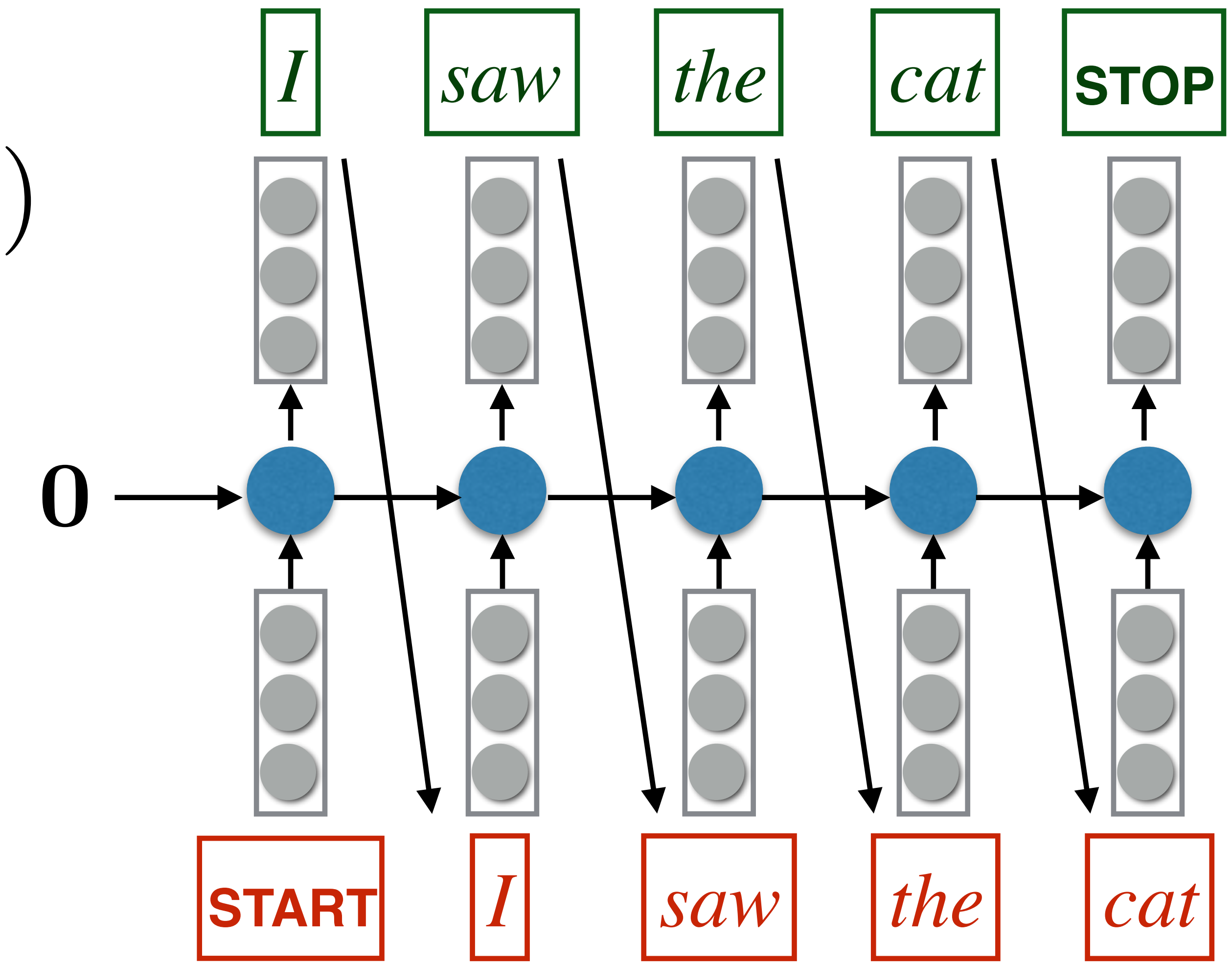


What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNN Language Models

$$\text{while } y_t \neq \text{STOP}$$
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$y_t \sim g(\mathbf{h}_t)$$
$$t \leftarrow t + 1$$



What is the probability of a sequence $\boldsymbol{y}$?

$$p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$$

RNNLMs for Structured Prediction

- Intuition $p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})$

RNNLMs for Structured Prediction

- Intuition $$\cancel{p(\boldsymbol{y}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i})}$$

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_i p(y_i \mid \boldsymbol{y}_{<i}, \boldsymbol{x})$$

Transition-Based Models

- Break the structure you want to build down into a sequence of structure-building operations (or **transitions**)

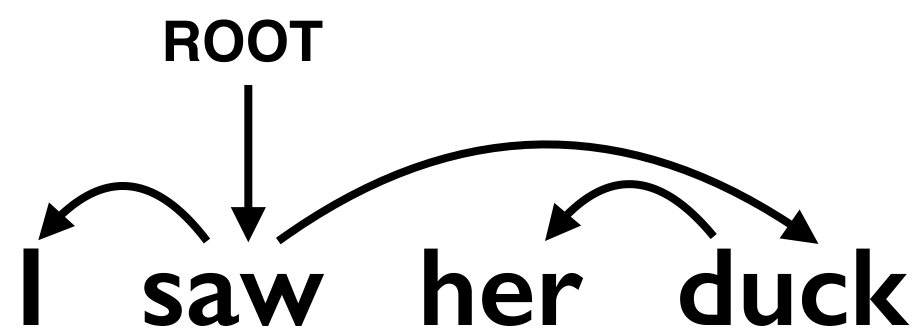
- sequence tagging can be done with a single operation:
  `ReadAndLabel(X)` - remove the next input symbol and label it with an X

  - more complicated structures (trees, graphs) require auxiliary data structures that are manipulated (more later)

Dependency parsing

I saw her duck

Dependency parsing



Transition-based parsing

- Build trees by pushing words ("**shift**") onto a stack and combing elements at the top of the stack into a syntactic constituent ("**reduce**")

- ***Given current stack and buffer of unprocessed words, what action should the algorithm take?***

- Widely used
  - Good accuracy
  - $O(n)$ runtime [much faster than other parsing algos]

Transition-based parsing

- There are actually perhaps 5 or 6 different “transition sets” for transition-based parsing (the one we are presenting is called “**arc standard**”)

- They use the stack and buffer in slightly different ways and may make predicting certain tree structures more or less difficult

- When designing your transition sets for your problem, keep in mind that there may be many possibilities

Stack	Buffer	Action
	I saw her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw		

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	

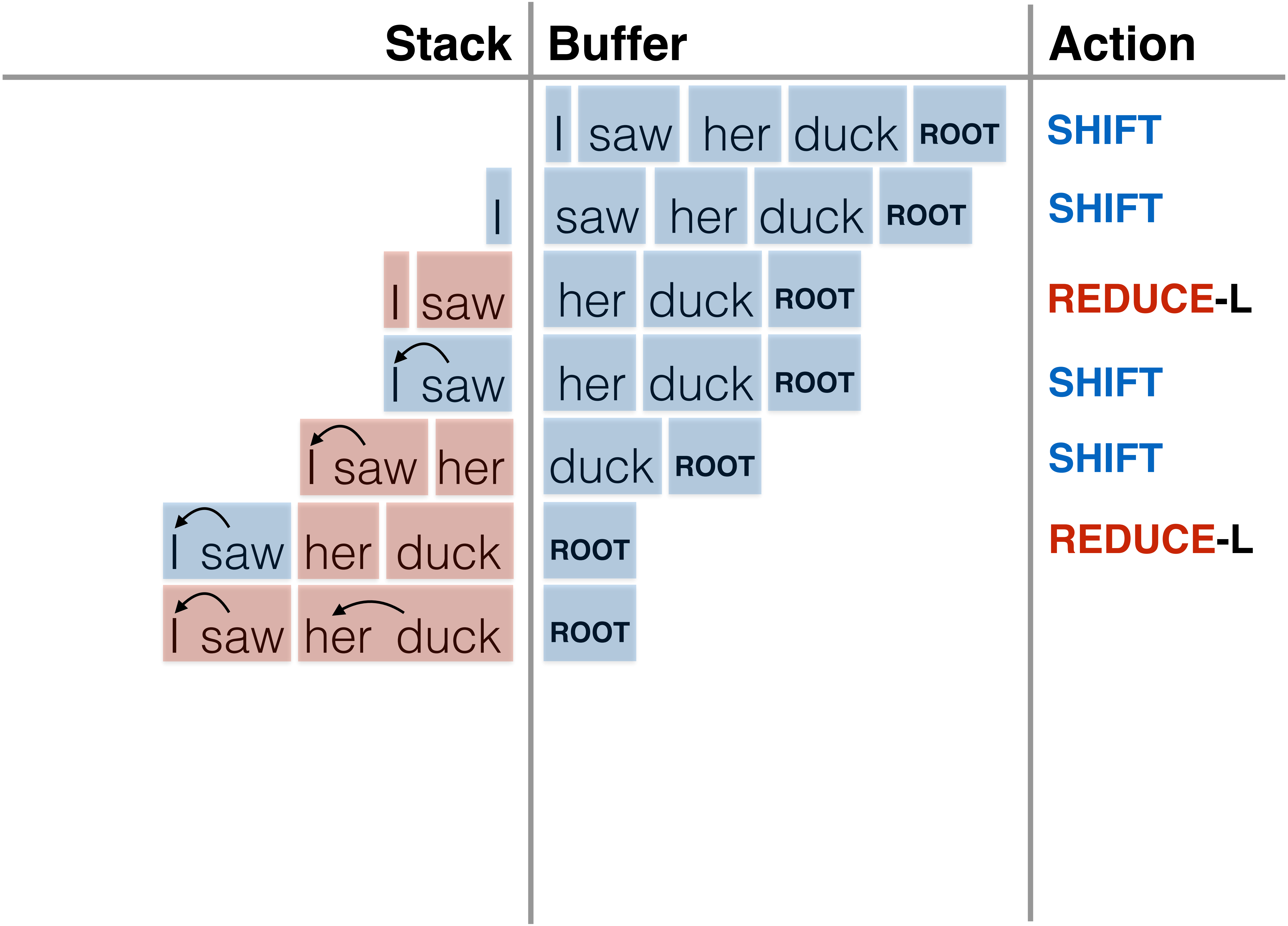
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	

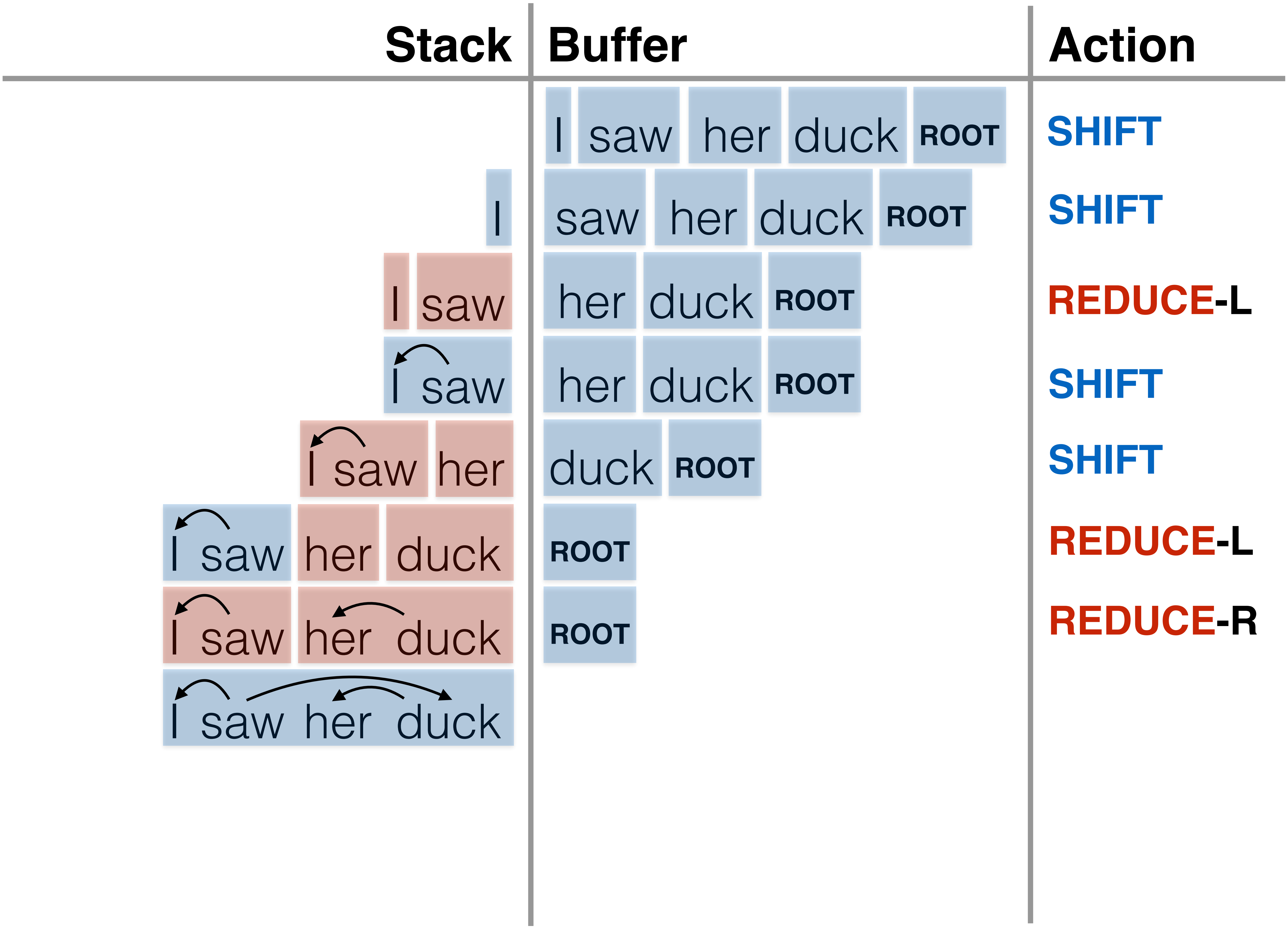
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L


Stack
Buffer
Action
I saw her duck ROOT
SHIFT
I
saw her duck ROOT
SHIFT
I saw
her duck ROOT
REDUCE-L
I saw
her duck ROOT
SHIFT
I saw her
duck ROOT
SHIFT
I saw her duck
ROOT
REDUCE-L
I saw her duck
ROOT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L
I saw her duck	ROOT	REDUCE-R



Stack
Buffer
Action
I saw her duck ROOT
SHIFT
I
saw her duck ROOT
SHIFT
I saw
her duck ROOT
REDUCE-L
I saw
her duck ROOT
SHIFT
I saw her
duck ROOT
SHIFT
I saw her duck
ROOT
REDUCE-L
I saw her duck
ROOT
REDUCE-R
I saw her duck

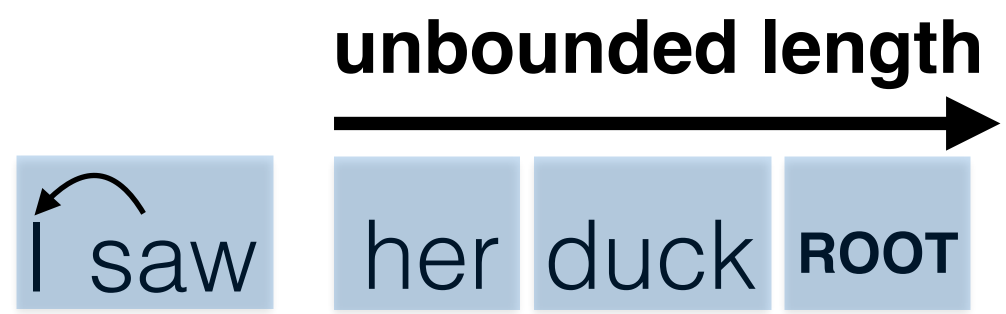
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L
I saw her duck	ROOT	REDUCE-R
I saw her duck	ROOT	SHIFT
I saw her duck ROOT		REDUCE-L
I saw her duck ROOT		

Making Predictions

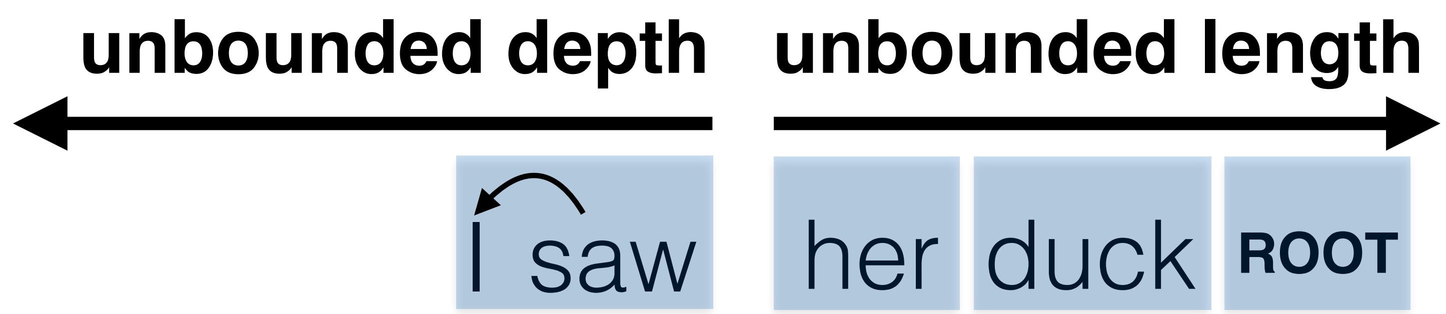
- In transition based models, you need to look at the current “state” of the algorithm and make a decision about what to do next
- The current state in sequence models is pretty simple
  - The things you’ve labeled
  - The labels you’ve produced
  - The unlabeled part of the string
- What about in trees?

Transition-based parsing
Challenges

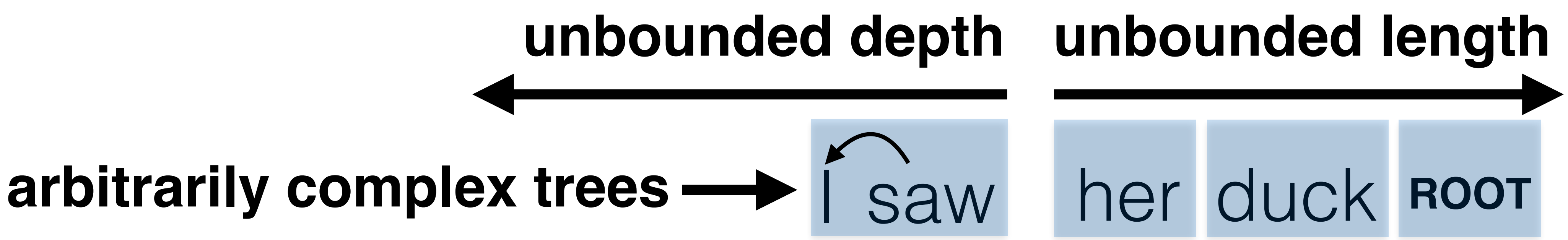
Transition-based parsing
Challenges



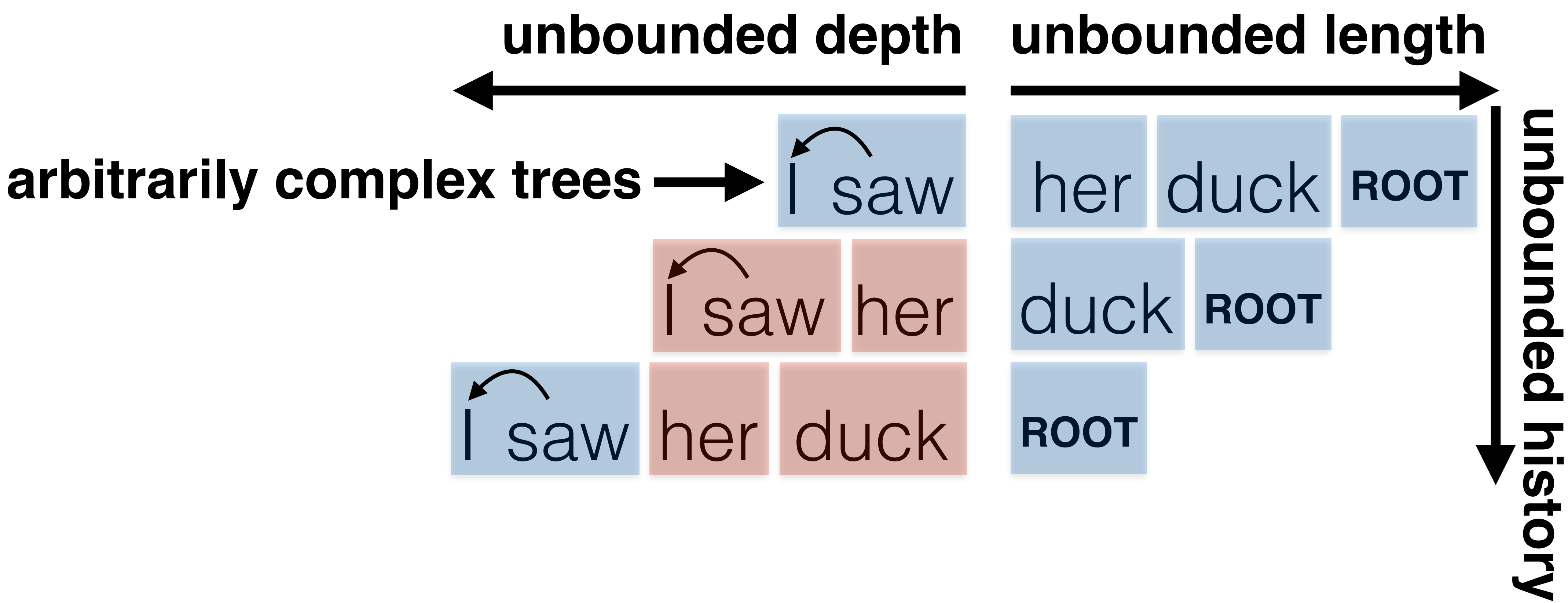
Transition-based parsing
Challenges



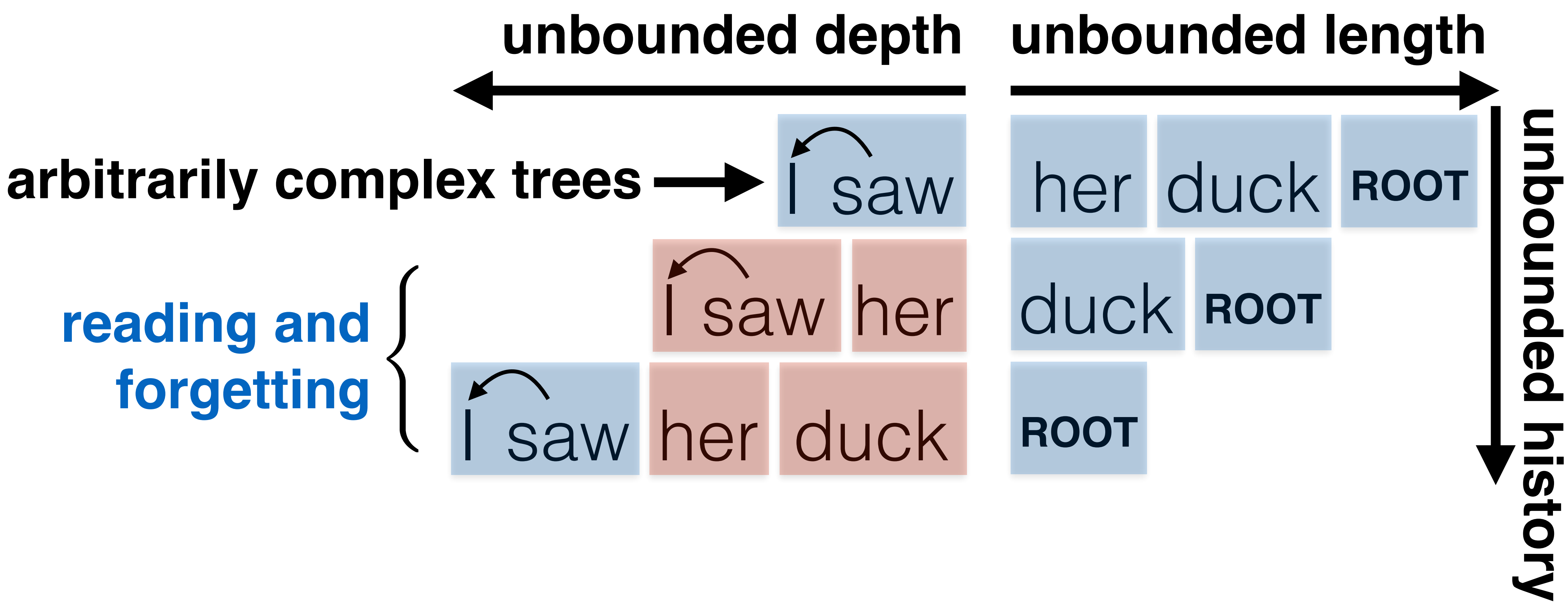
Transition-based parsing
Challenges



Transition-based parsing
Challenges



Transition-based parsing
Challenges



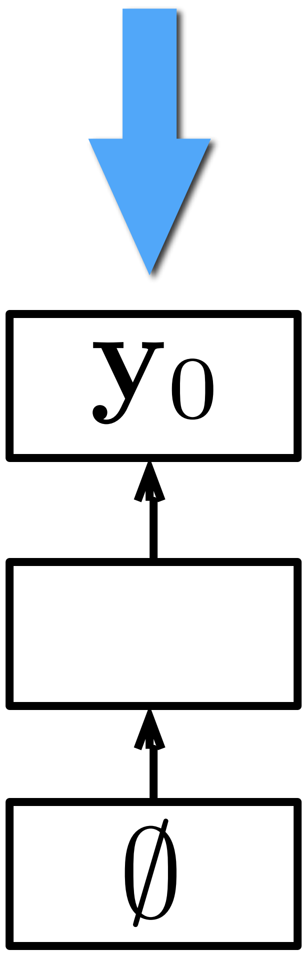
Transition-based parsing
Solutions

- Use a new variant of LSTMs—**stack LSTMs**—to embed buffer, stack, and history of actions
  - Embeddings are sensitive to full lookahead, full stack contents, and full history of actions
  - Incremental construction of parser state embedding means **runtime remains linear**

Transition-based parsing
Stack LSTMs

- Augment LSTM with a **stack pointer**
- Two constant-time operations
  - **Push** - read input, add to top of stack
  - **Pop** - move stack pointer back
- A **summary** of stack contents is obtained by accessing the output of the LSTM at location of the stack pointer

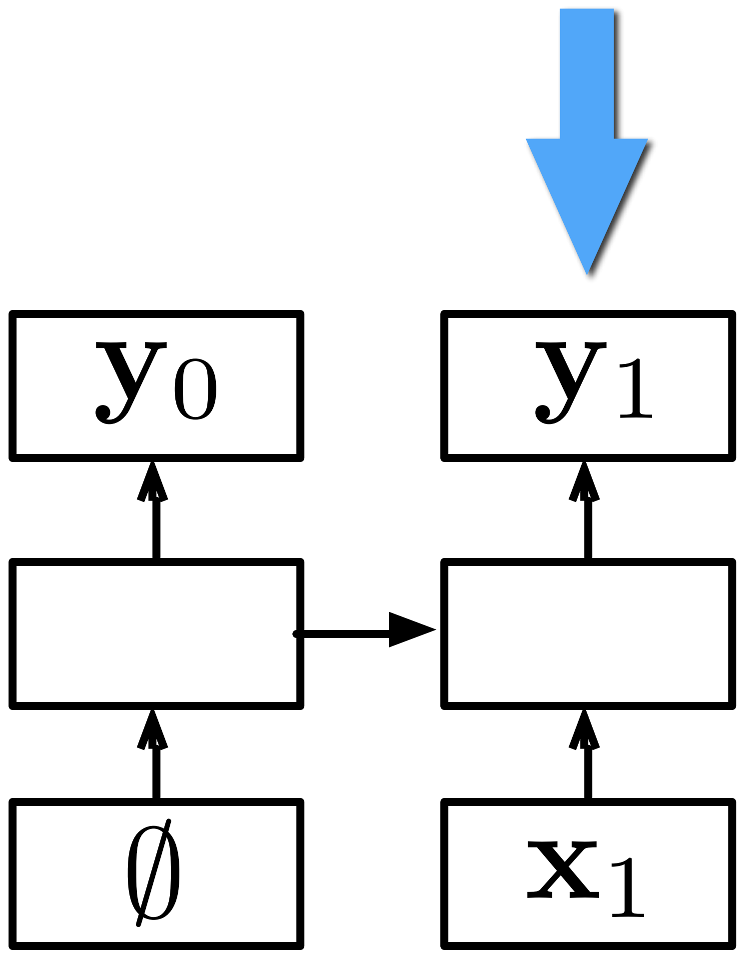
Transition-based parsing
Stack LSTMs



PUSH

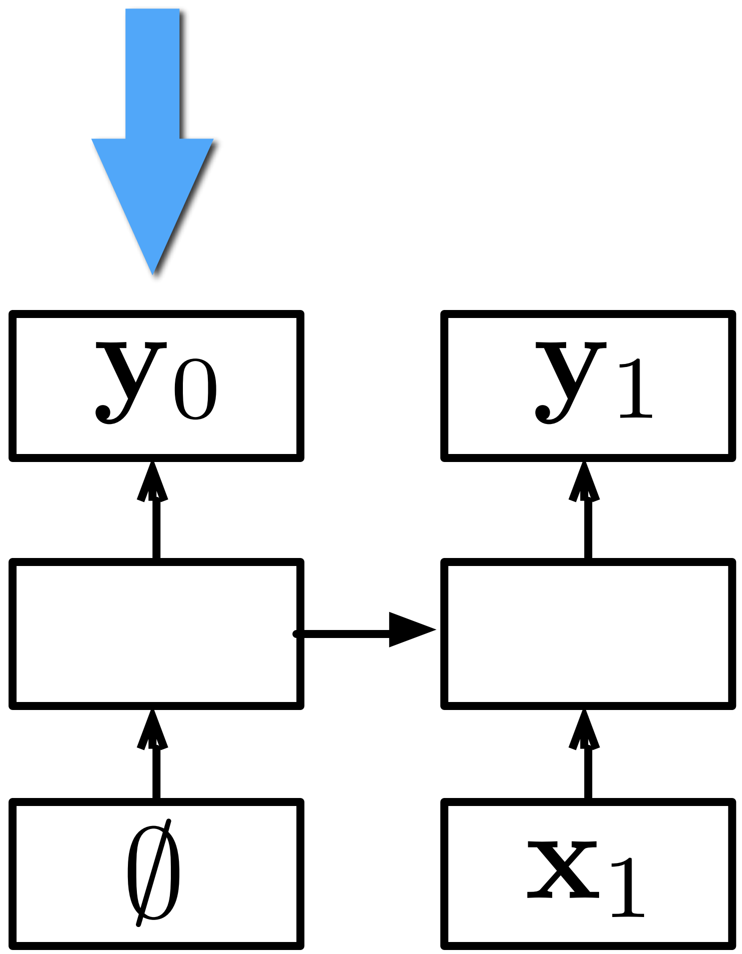
Transition-based parsing

Stack LSTMs

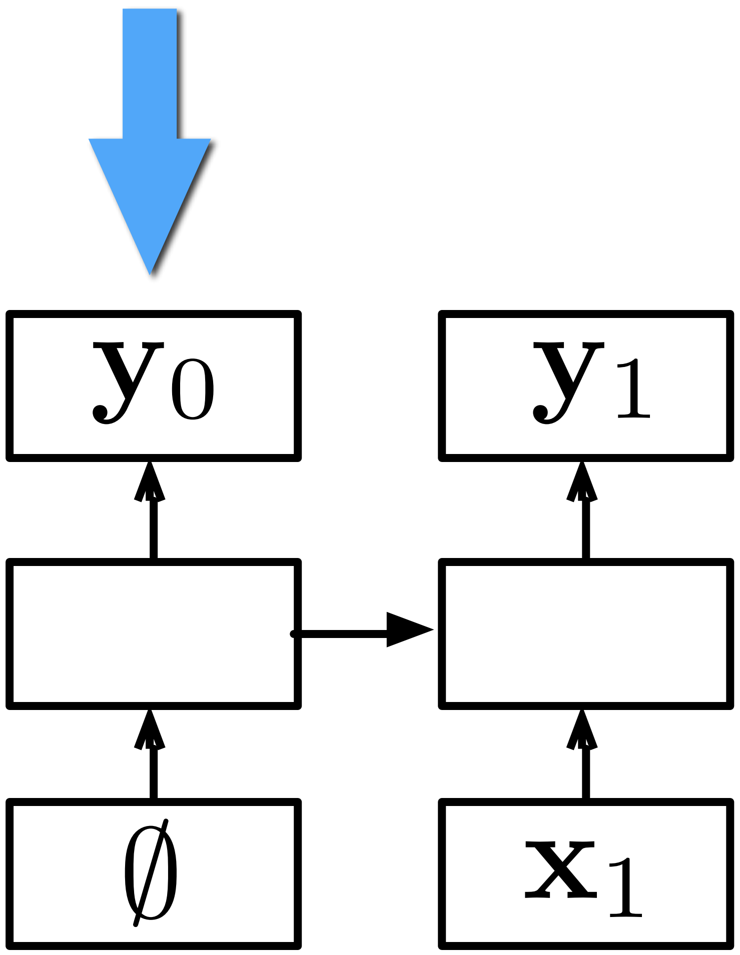


POP

Transition-based parsing
Stack LSTMs

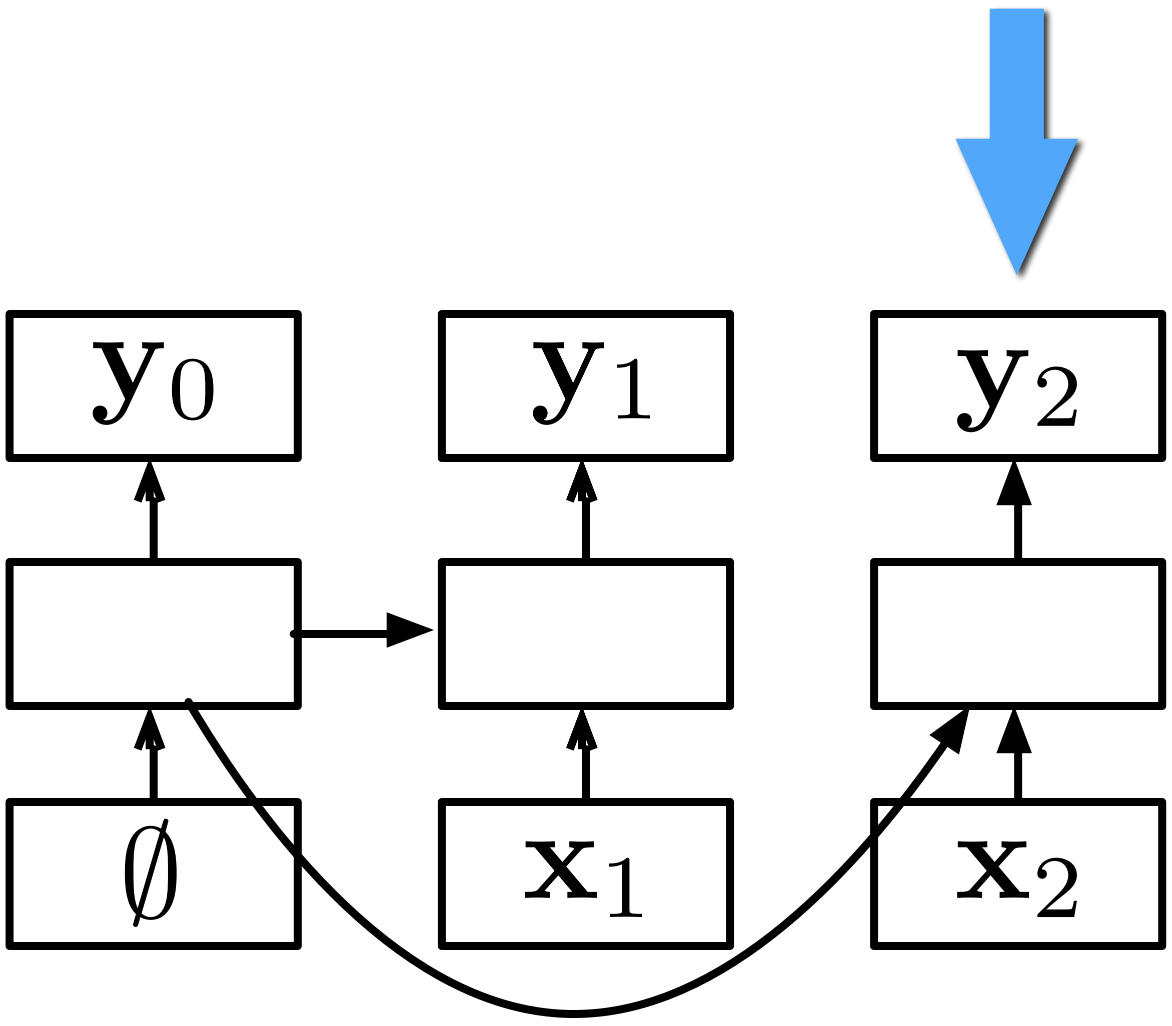


Transition-based parsing
Stack LSTMs



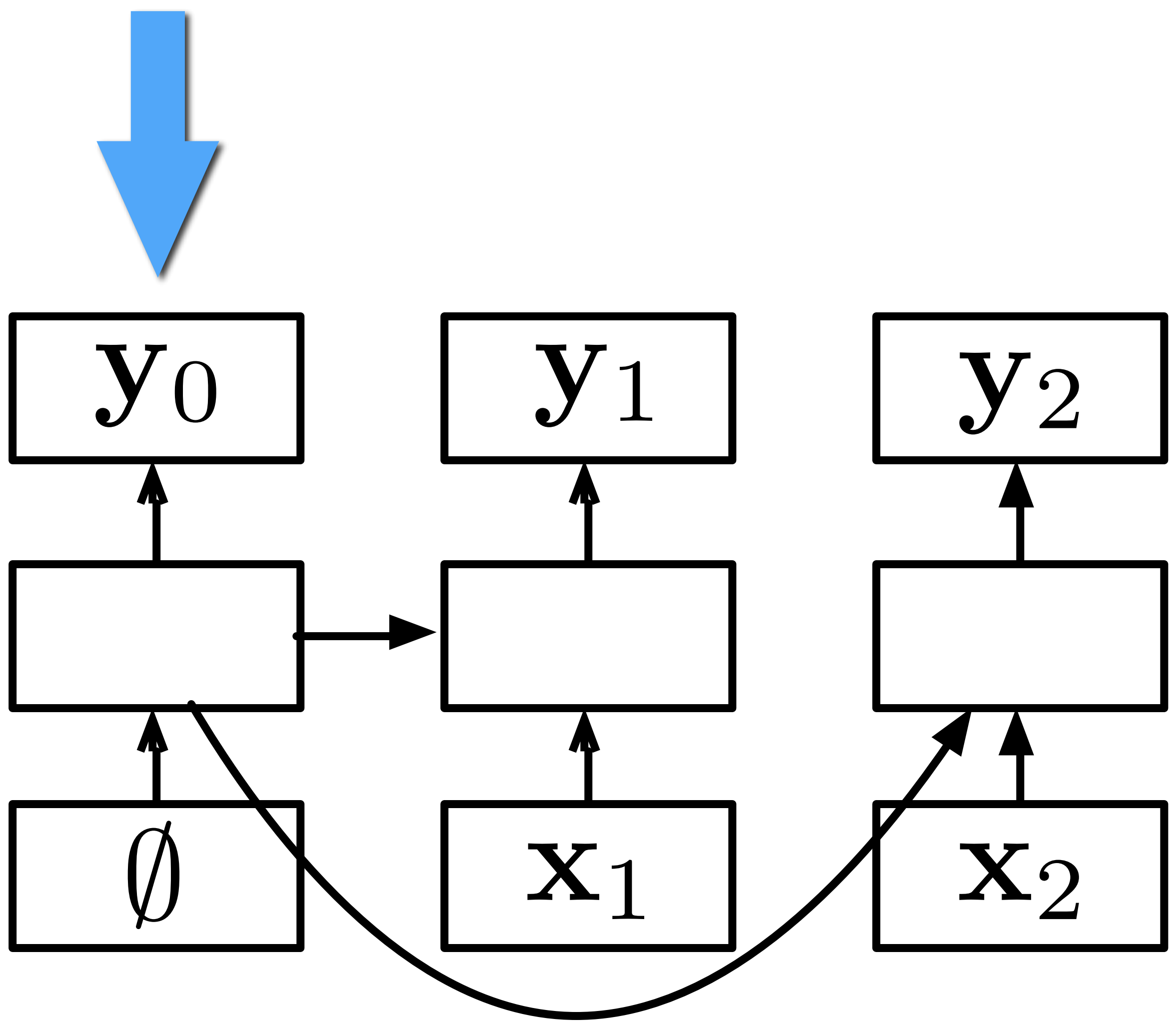
PUSH

Transition-based parsing
Stack LSTMs

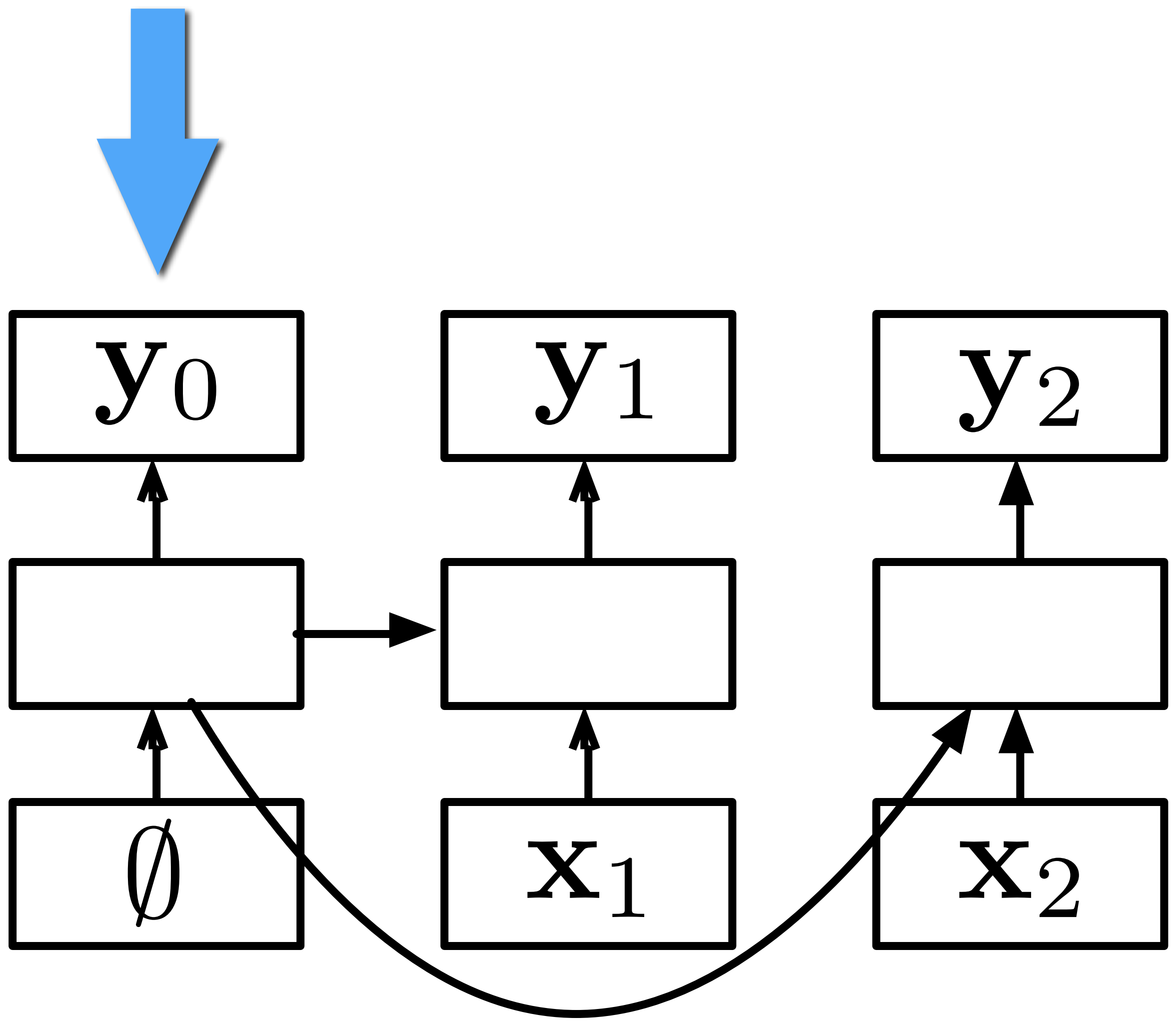


POP

Transition-based parsing
Stack LSTMs

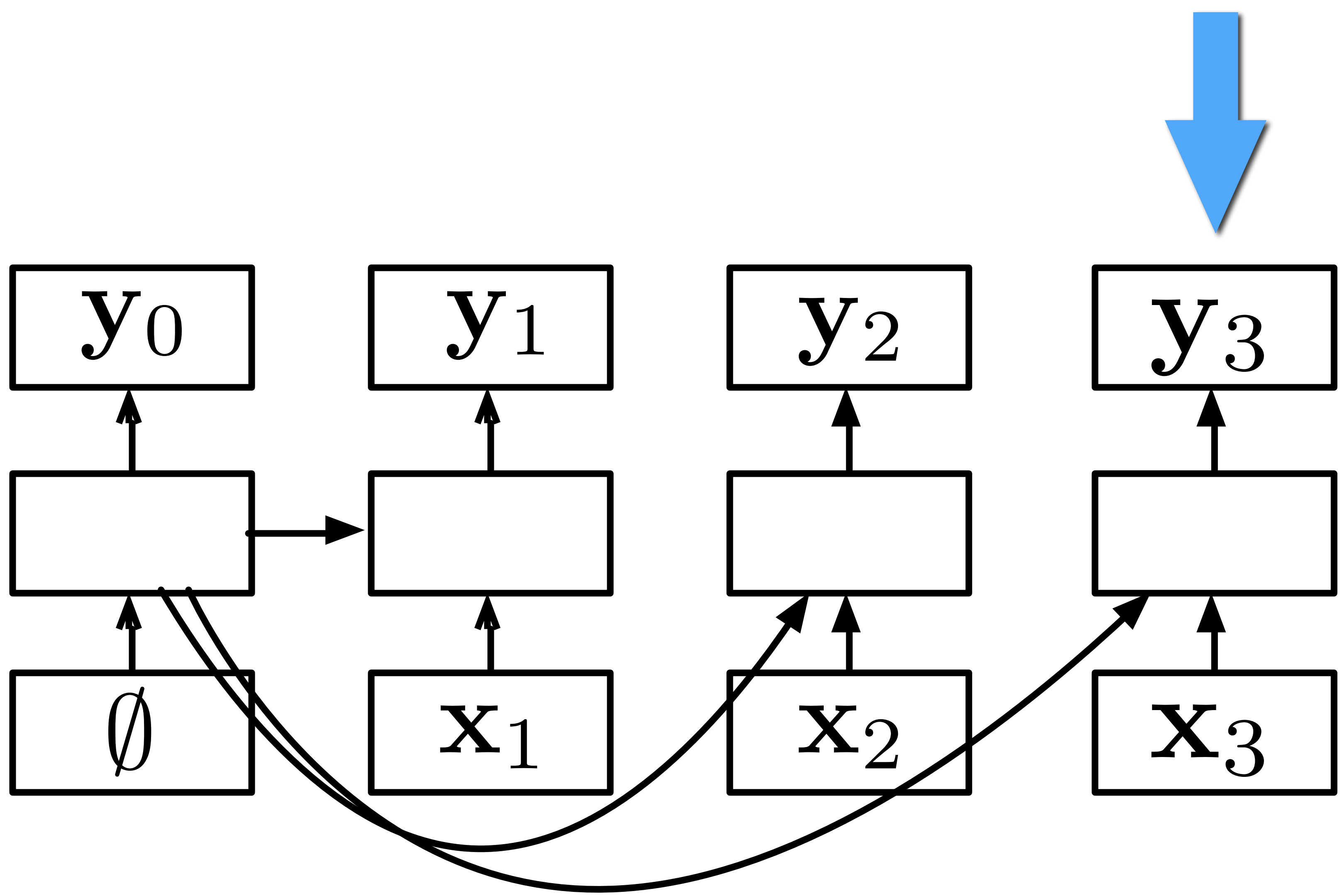


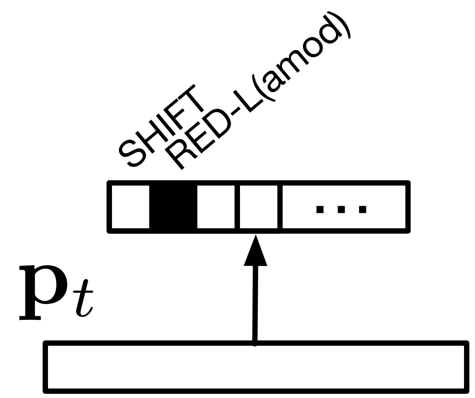
Transition-based parsing
Stack LSTMs

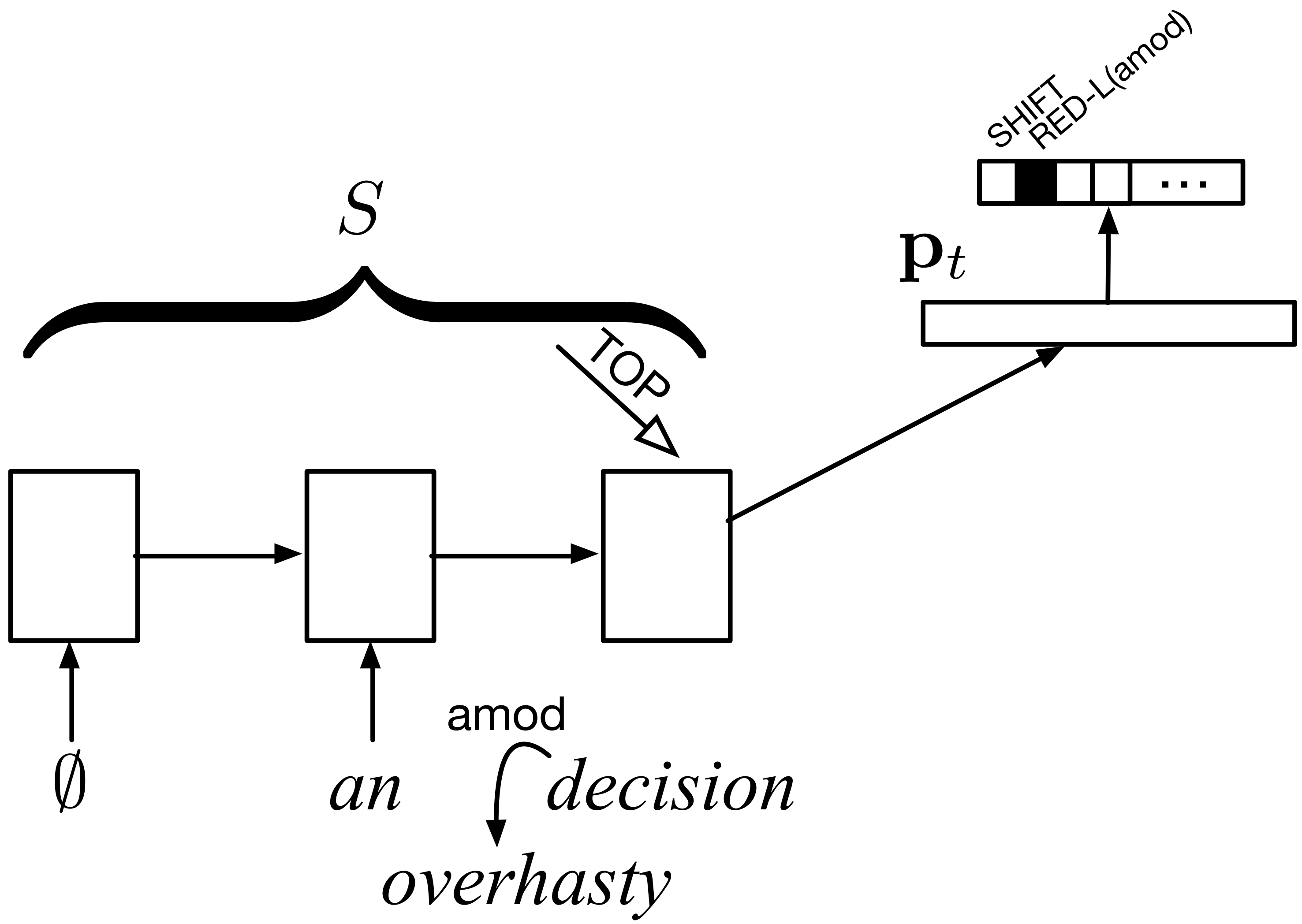


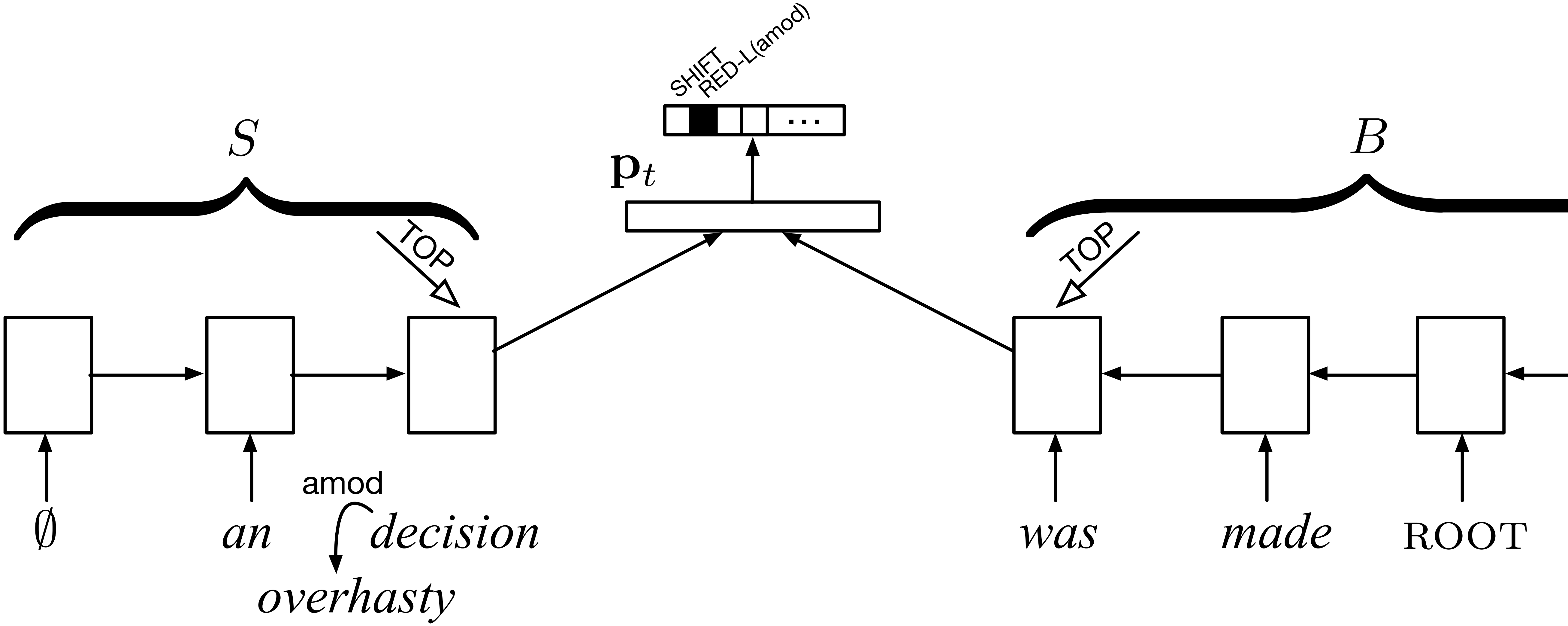
PUSH

Transition-based parsing
Stack LSTMs

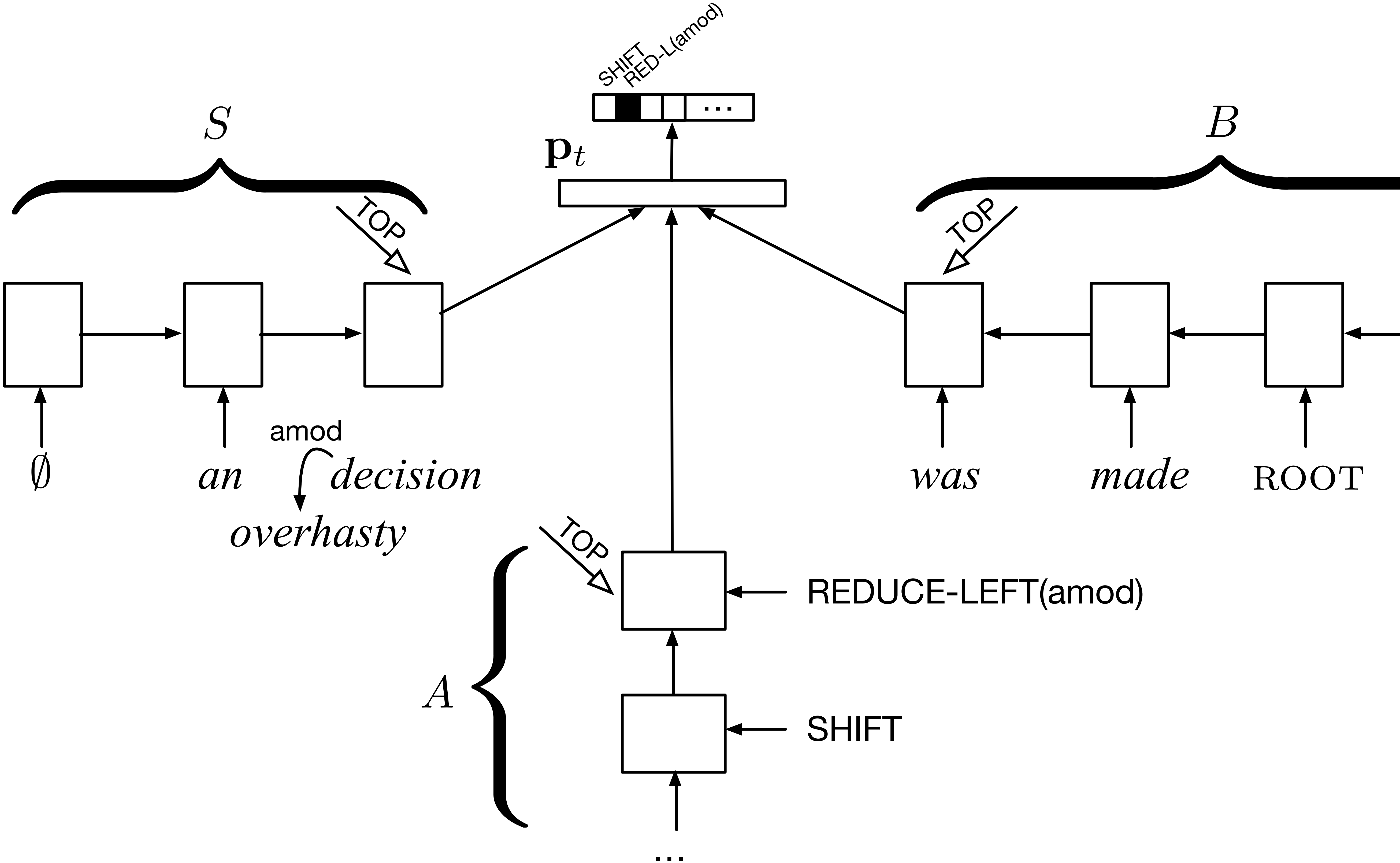



SHIFT
RED-L(amod)
...
$\mathbf{p}_t$


SHIFT
RED-L(amod)
...
$\mathbf{p}_t$
S
TOP
$\emptyset$
an
amod
decision
overhasty

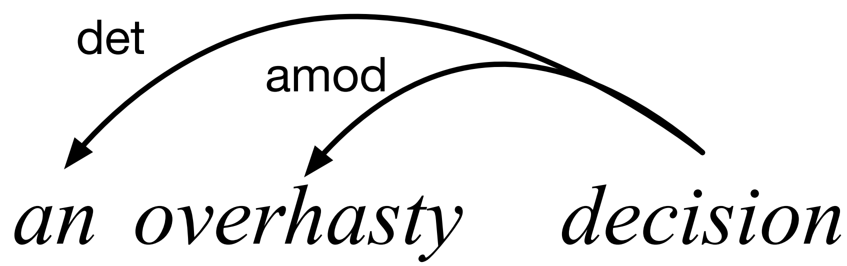


SHIFT
RED-L(amod)
...
$\mathbf{p}_t$
S
B
TOP
TOP
$\emptyset$
an
amod
decision
overhasty
was
made
ROOT

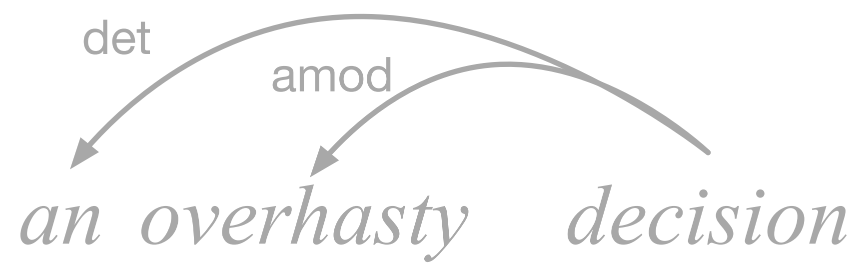


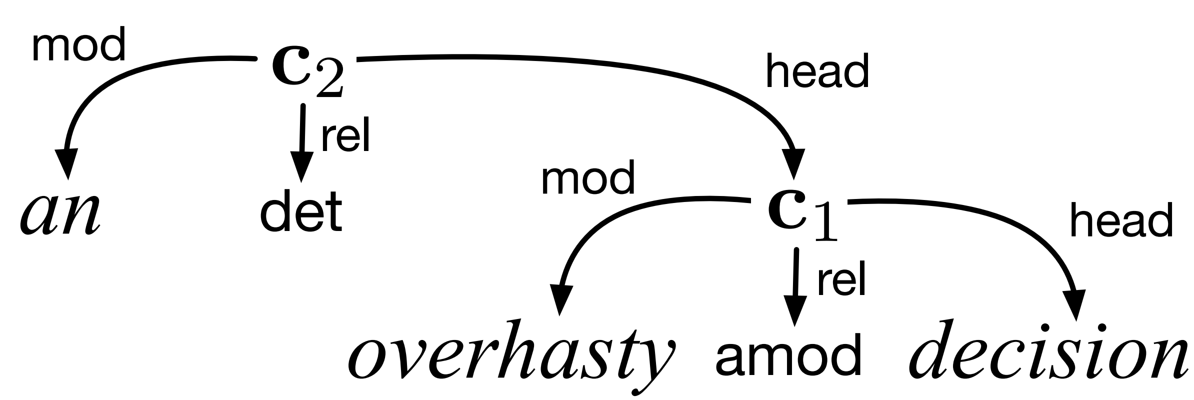
SHIFT
RED-L(amod)
$\mathbf{p}_t$
S
B
TOP
TOP
TOP
$\emptyset$
an
amod
decision
overhasty
was
made
ROOT
REDUCE-LEFT(amod)
SHIFT
A
...

Representing Tree(let)s



Representing Tree(let)s





Inference

$$\begin{aligned}\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{y}} \prod_i p(y_i \mid \boldsymbol{y}_{<i}, \boldsymbol{x})\end{aligned}$$

RNNs never forget anything! Decoding is difficult.

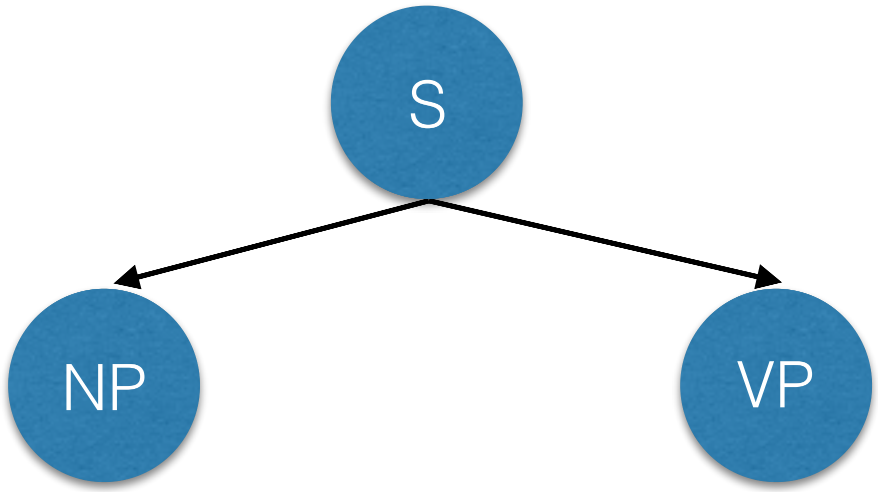
- **Greedy left-to-right decoding**
- **Beam search**
- **Particle filtering**

	Development		Test	
	UAS	LAS	UAS	LAS
S-LSTM	**93.2**	**90.9**	**93.1**	**90.9**
−POS	93.1	90.4	92.7	90.3
−pretraining	92.7	90.4	92.4	90.0
−composition	92.7	89.9	92.2	89.6
S-RNN	92.8	90.4	92.3	90.1
C&M (2014)	92.2	89.7	91.8	89.6

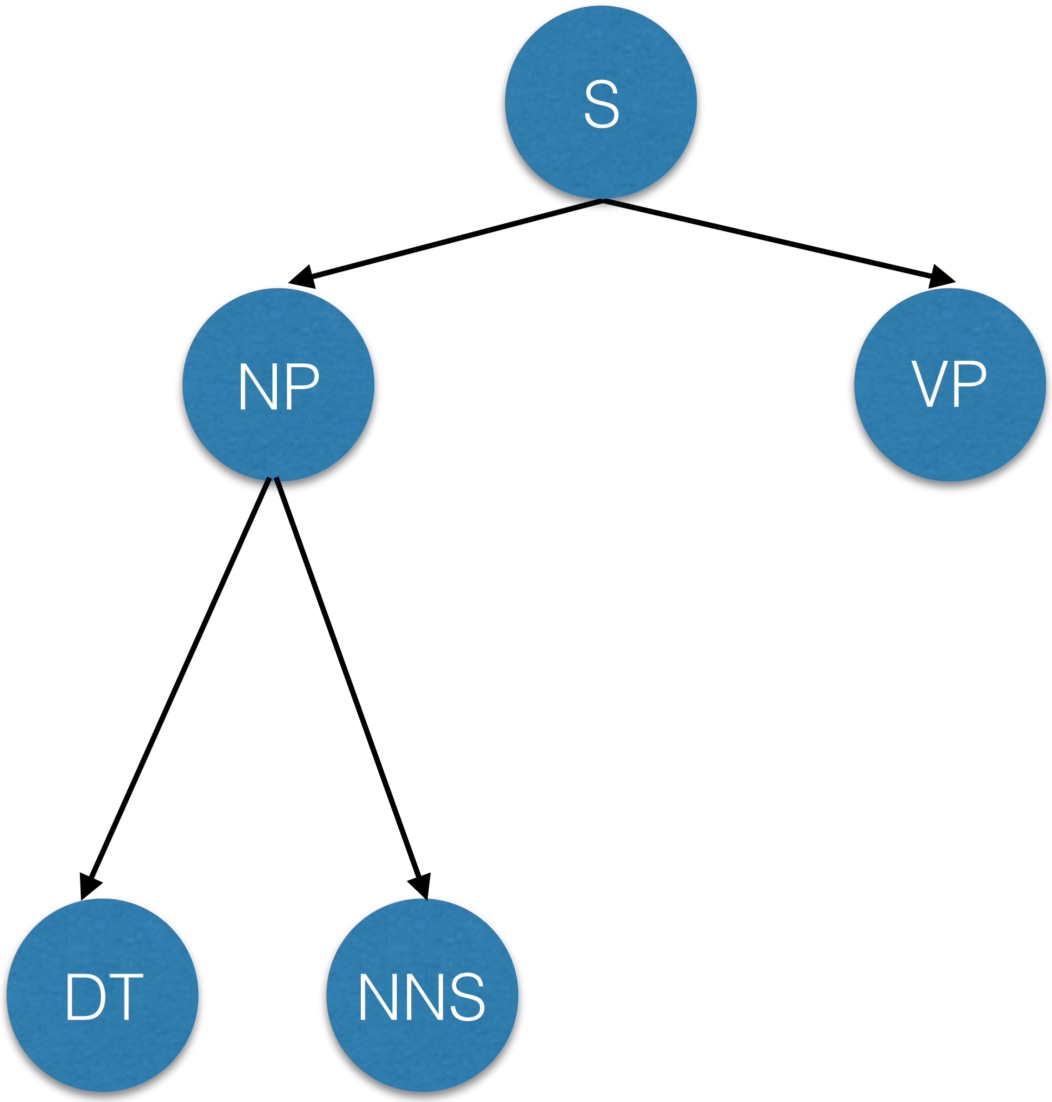
Other examples

- Constituency parsing
    - both top-down and bottom-up “unrollings” exist
- bottom-up
    - **shift** behaves as it did before
    - **reduce** builds a unary or binary constituent, also takes a label type (VP, NP, …)
- top-down
    - addition of a new operation: **NT**

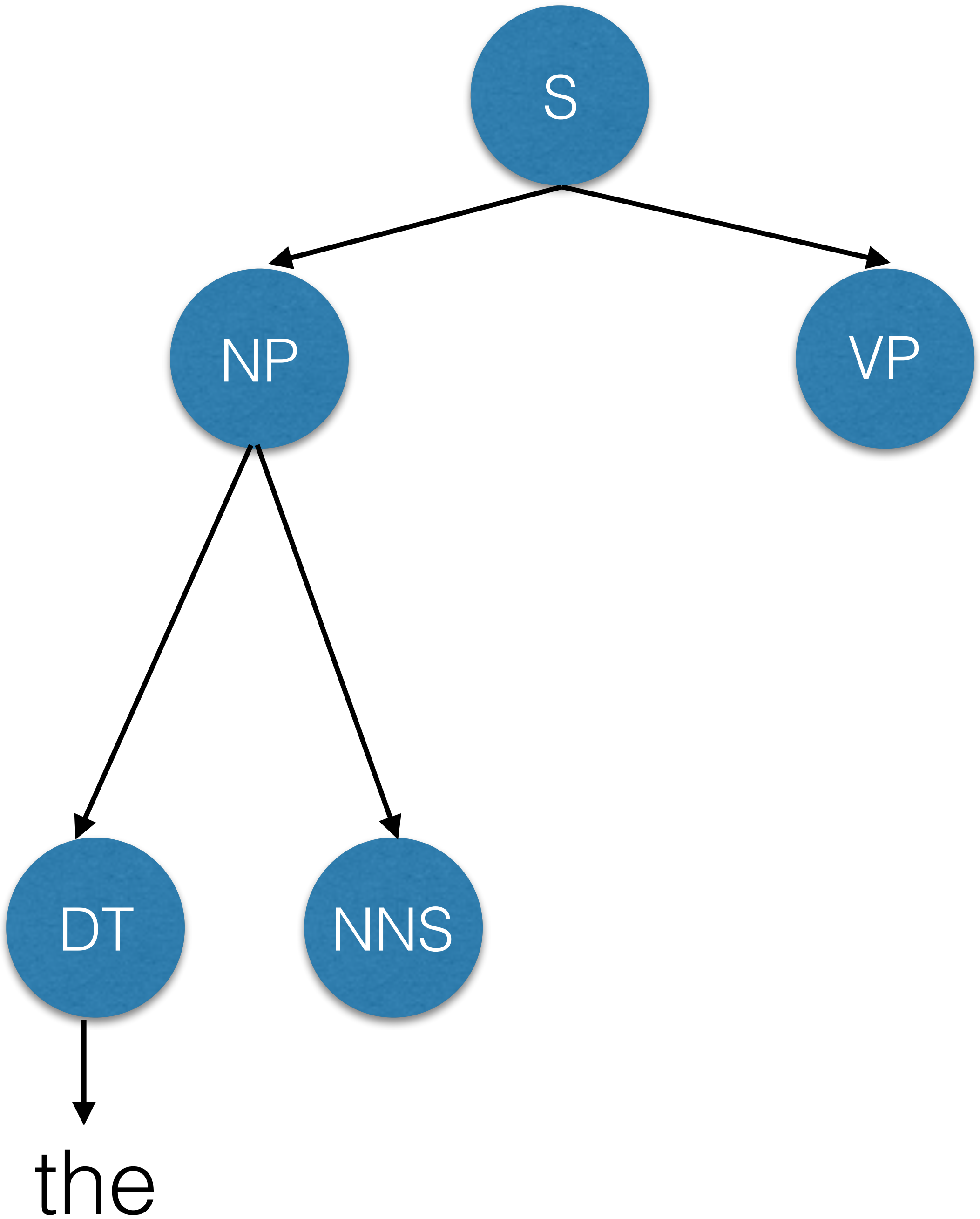
$$p(\mathrm{S})$$



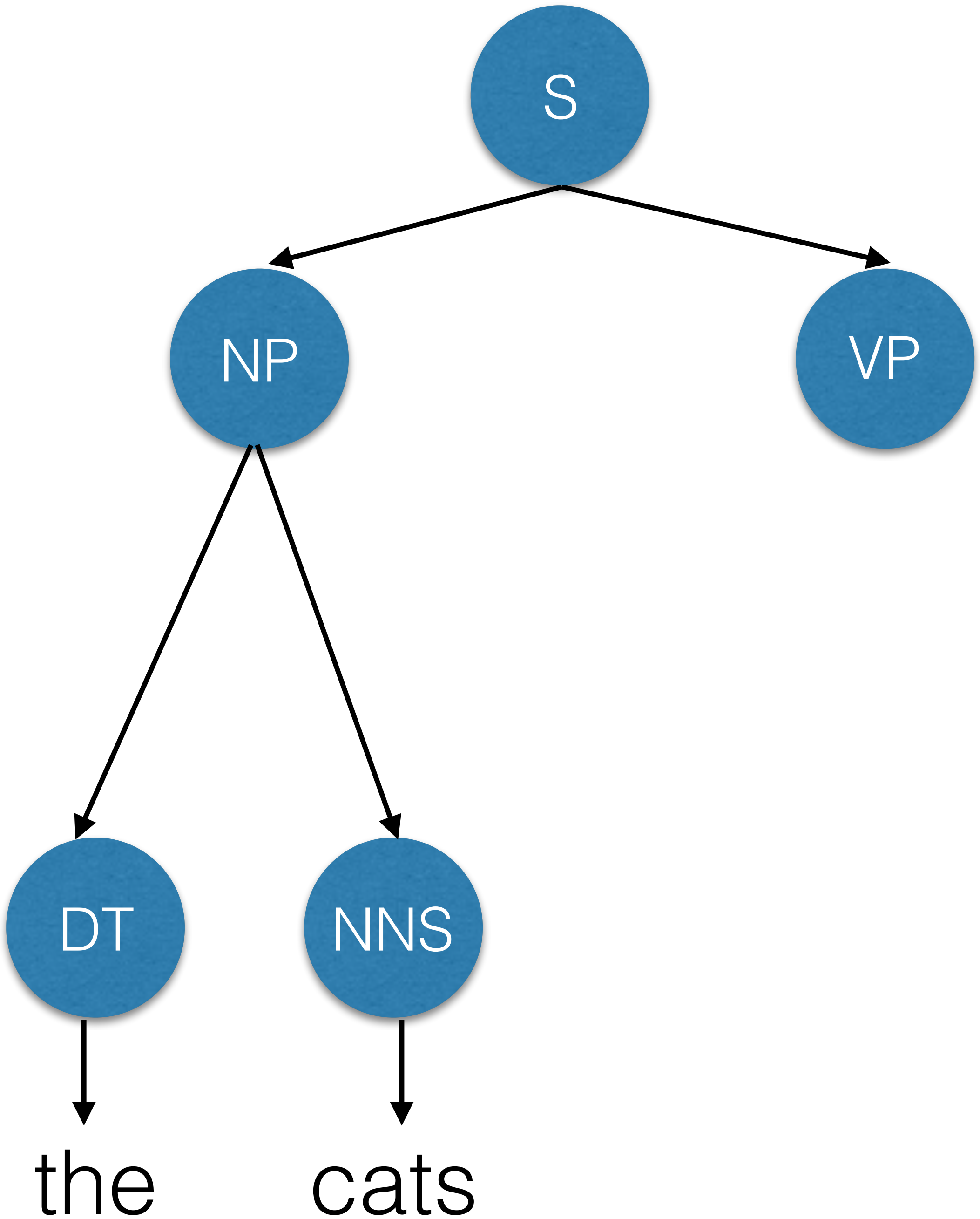
$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$



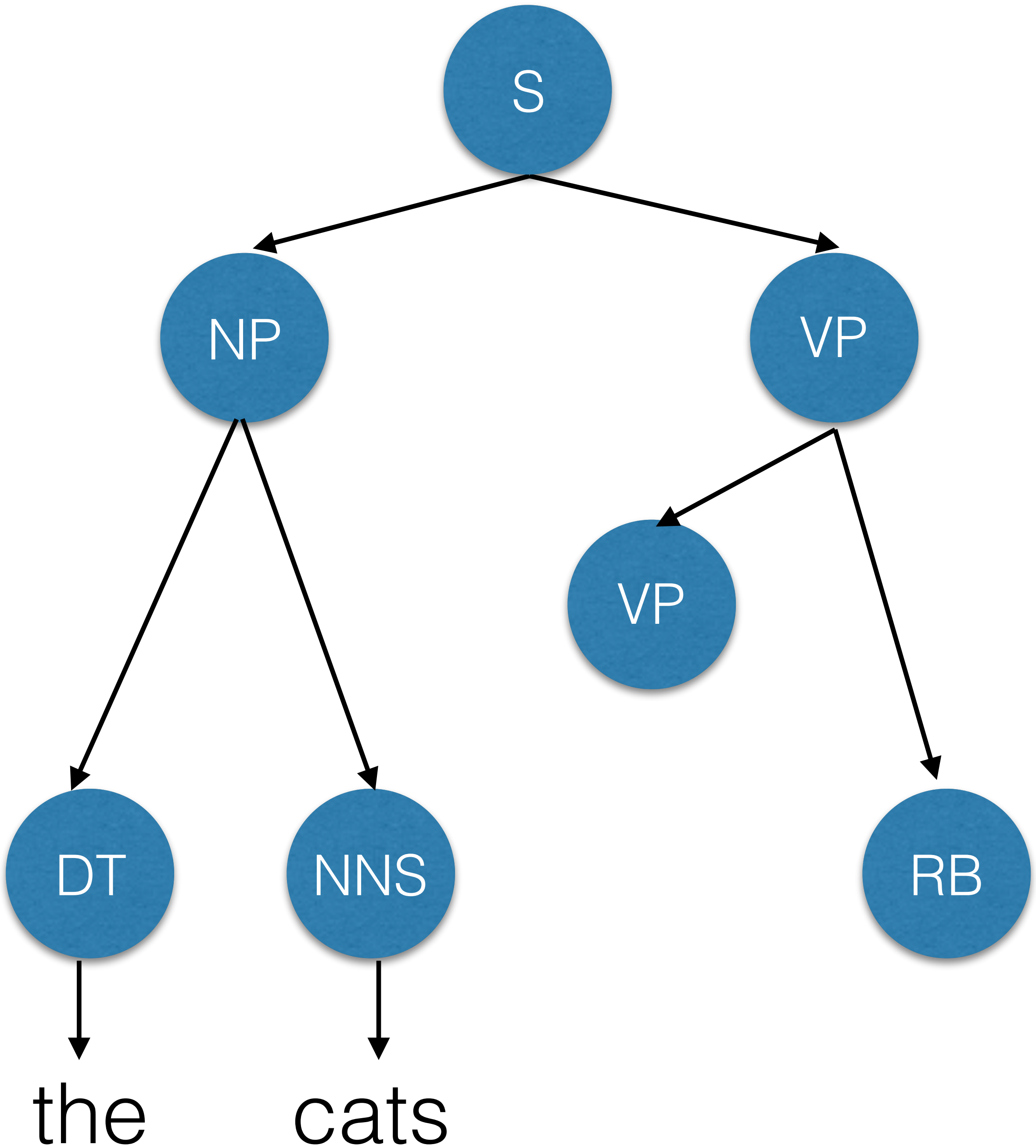
$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\textit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$



$p(\mathrm{S})$

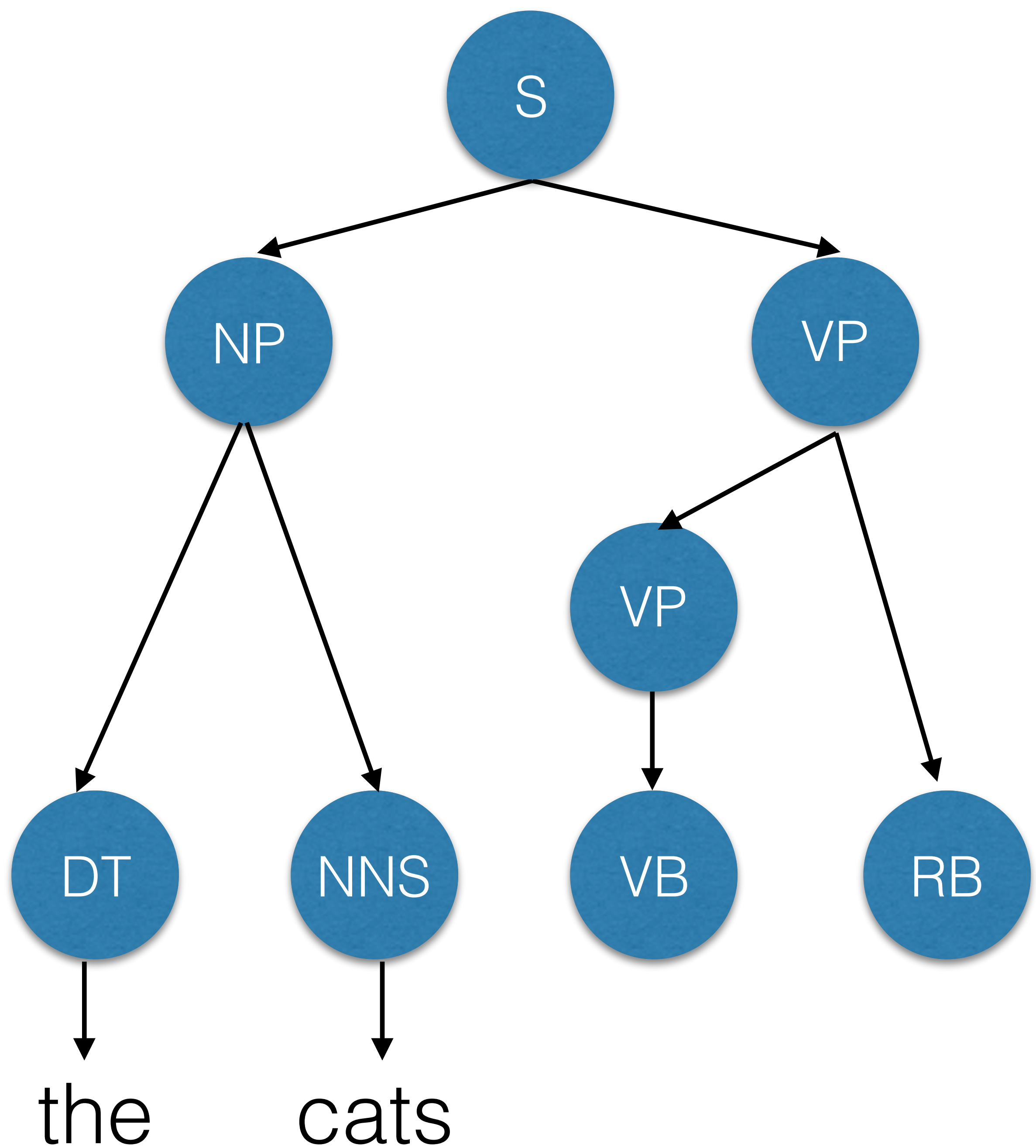
$p(\mathrm{NP\ VP} \mid \mathrm{S})$

$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$

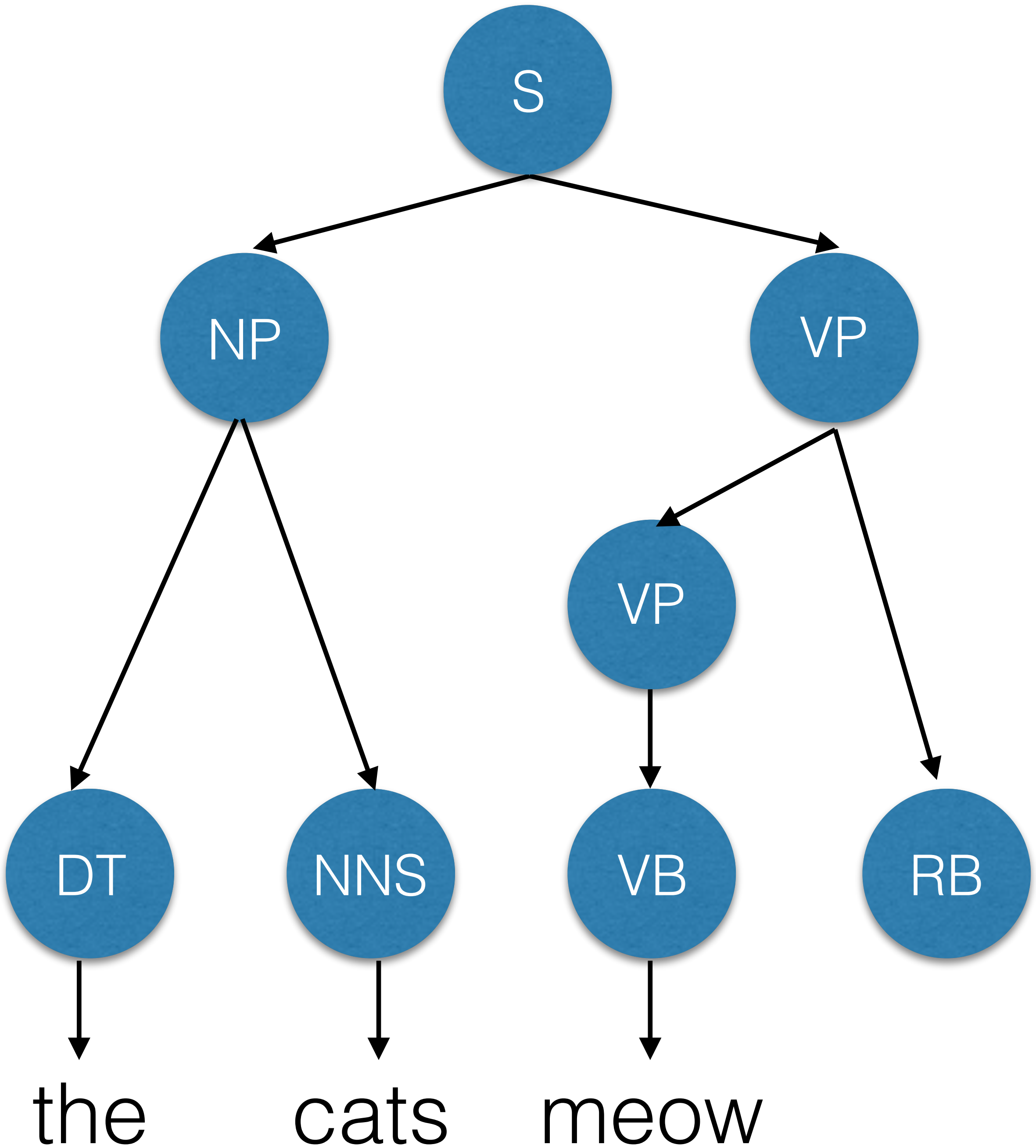
$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$

$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$

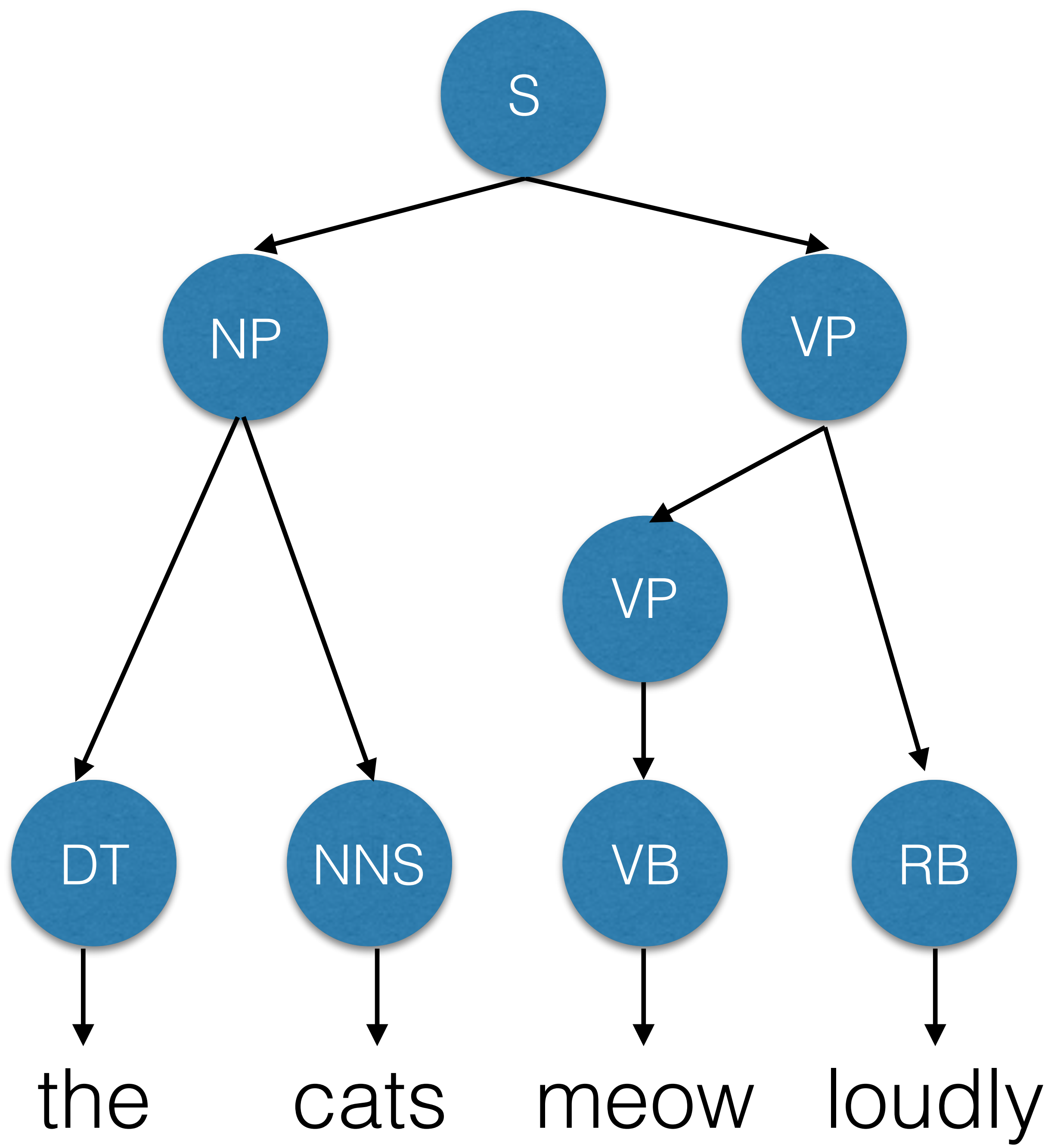
$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$

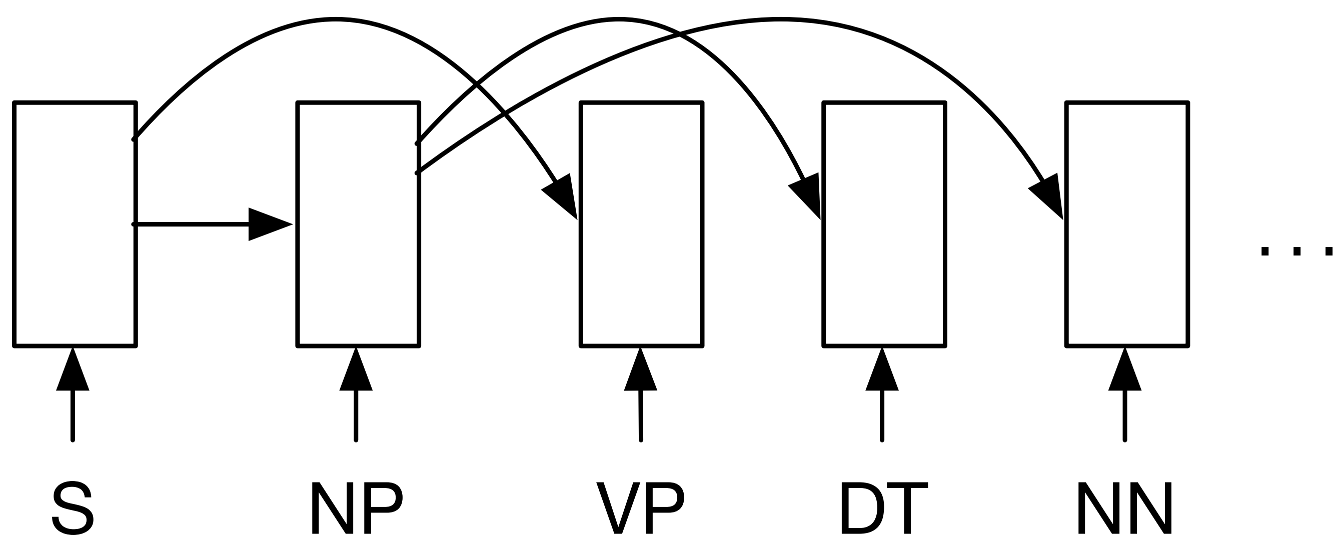


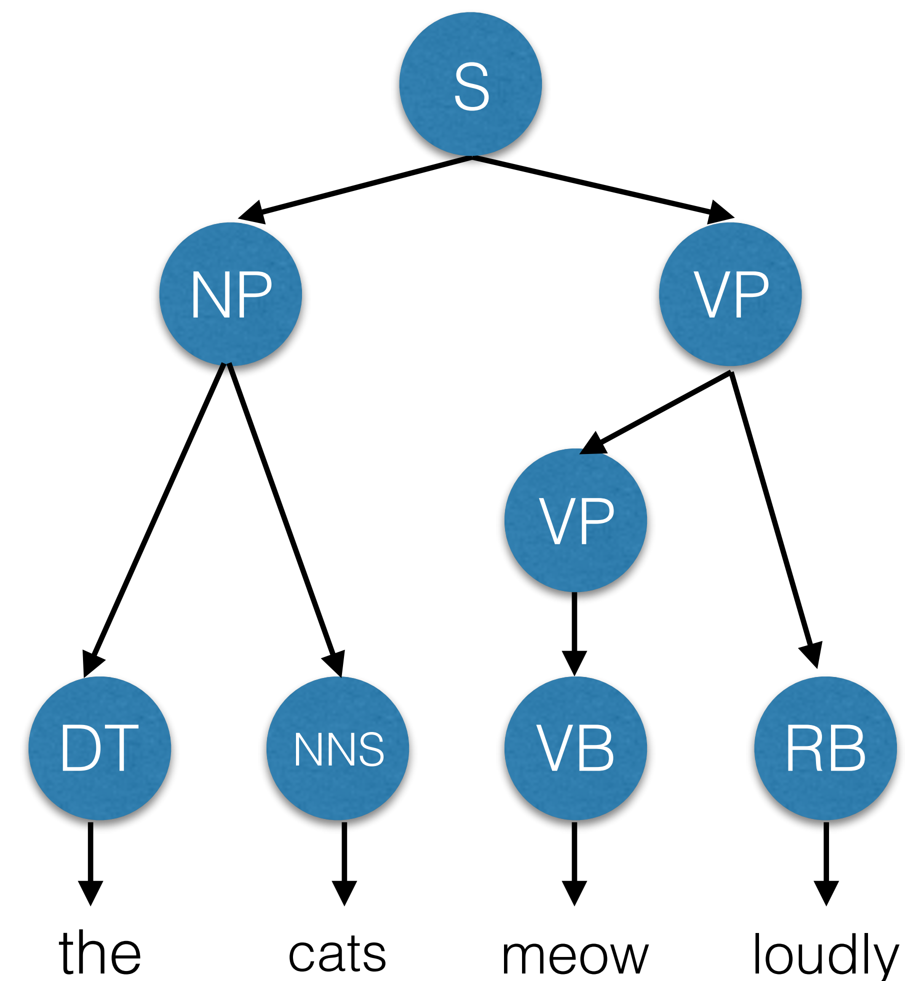
$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$
$$p(\mathit{meow} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP}, \mathrm{VB})$$



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$
$$p(\mathit{meow} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP}, \mathrm{VB})$$
$$p(\mathit{loudly} \mid \mathrm{S}, \mathrm{VP}, \mathrm{RB})$$

Top-Down Tree RNN

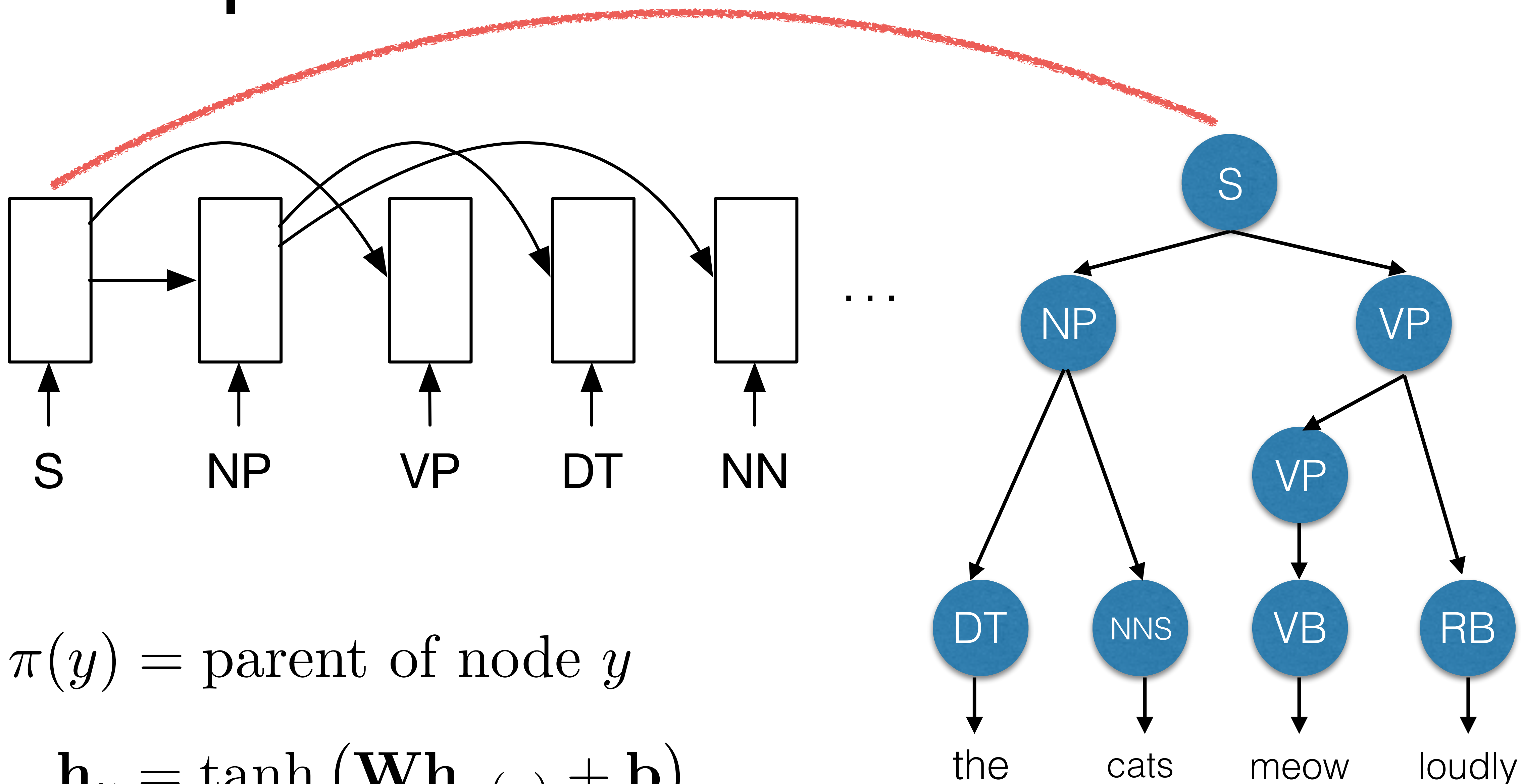




$$\pi(y) = \text{parent of node } y$$

$$\mathbf{h}_y = \tanh\left(\mathbf{W}\mathbf{h}_{\pi(y)} + \mathbf{b}\right)$$

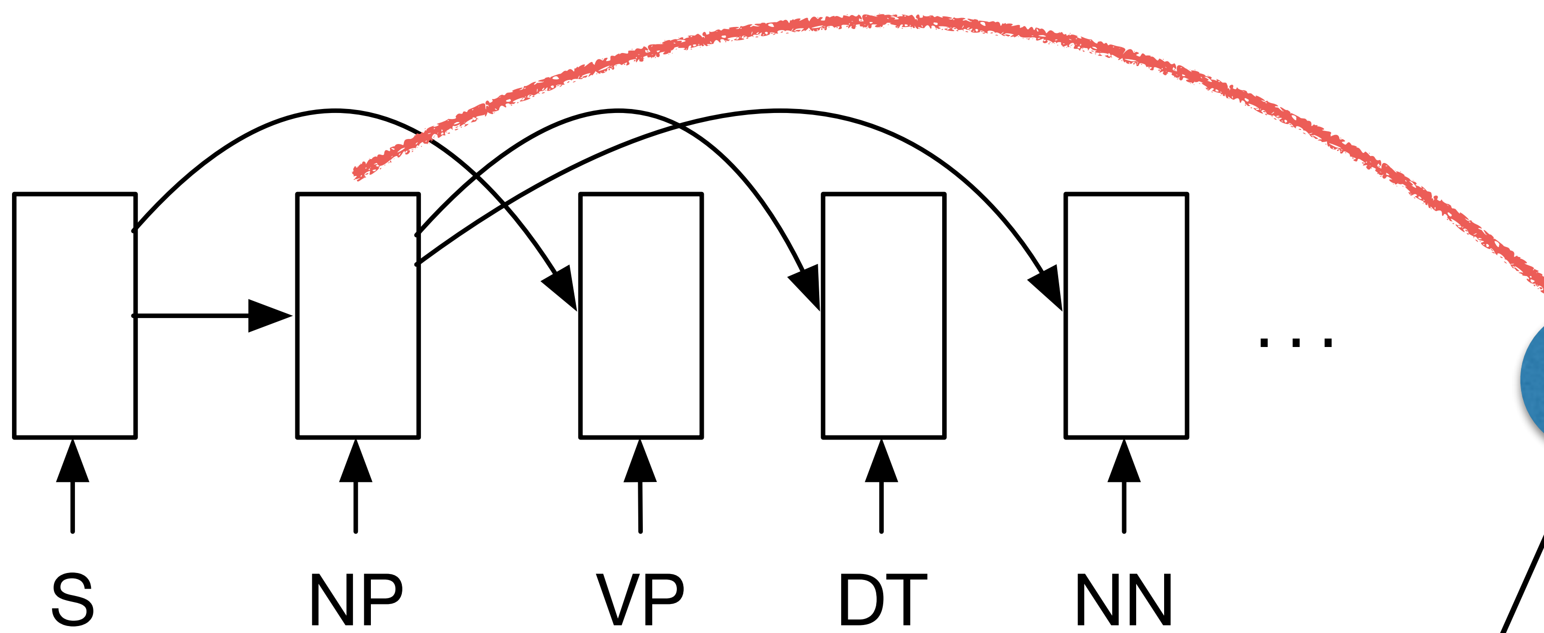
Top-Down Tree RNN

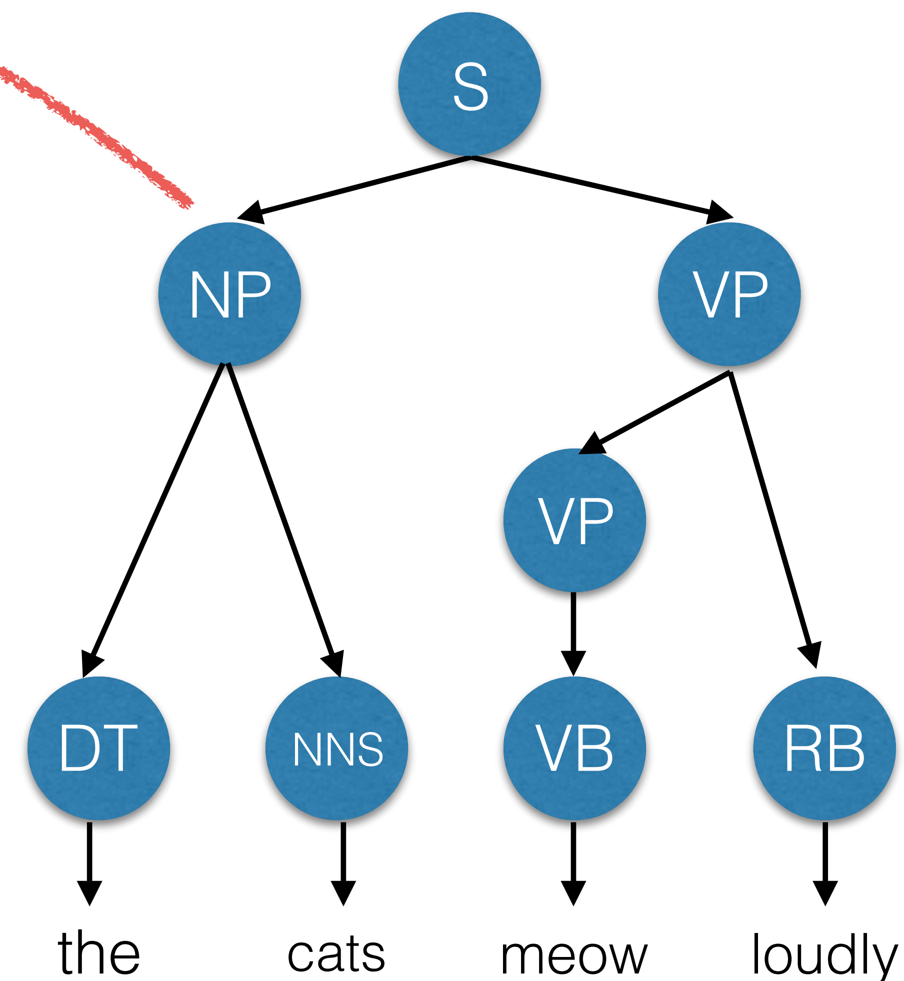


$$\pi(y) = \text{parent of node } y$$

$$\mathbf{h}_y = \tanh\left(\mathbf{W}\mathbf{h}_{\pi(y)} + \mathbf{b}\right)$$

Top-Down Tree RNN

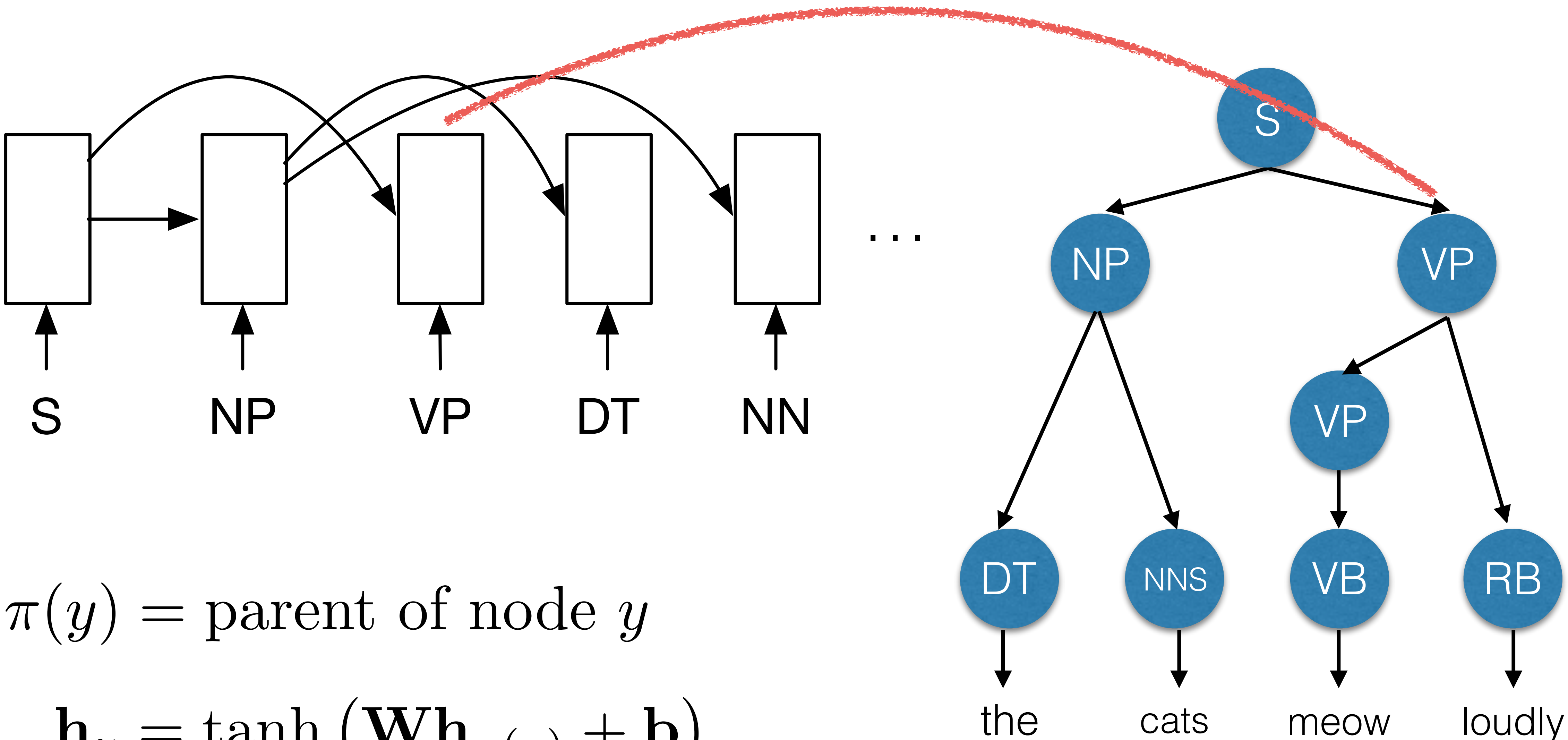




$$\pi(y) = \text{parent of node } y$$

$$\mathbf{h}_y = \tanh\left(\mathbf{W}\mathbf{h}_{\pi(y)} + \mathbf{b}\right)$$

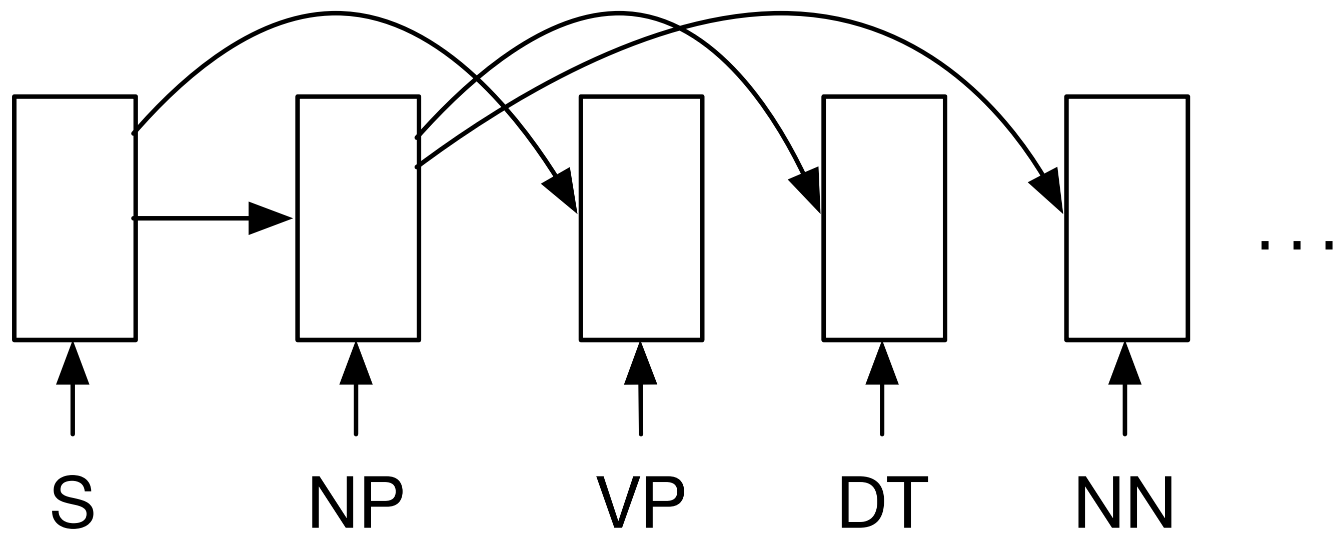
Top-Down Tree RNN

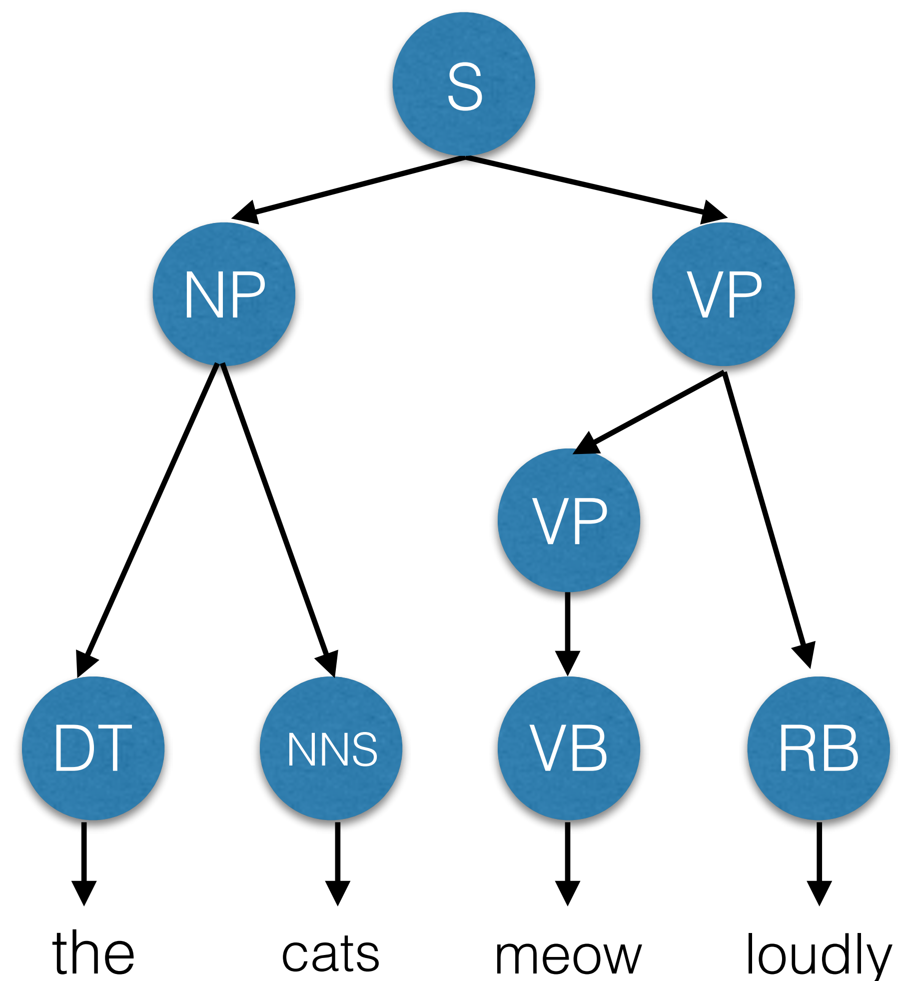


$$\pi(y) = \text{parent of node } y$$

$$\mathbf{h}_y = \tanh\left(\mathbf{W}\mathbf{h}_{\pi(y)} + \mathbf{b}\right)$$

Top-Down Tree RNN



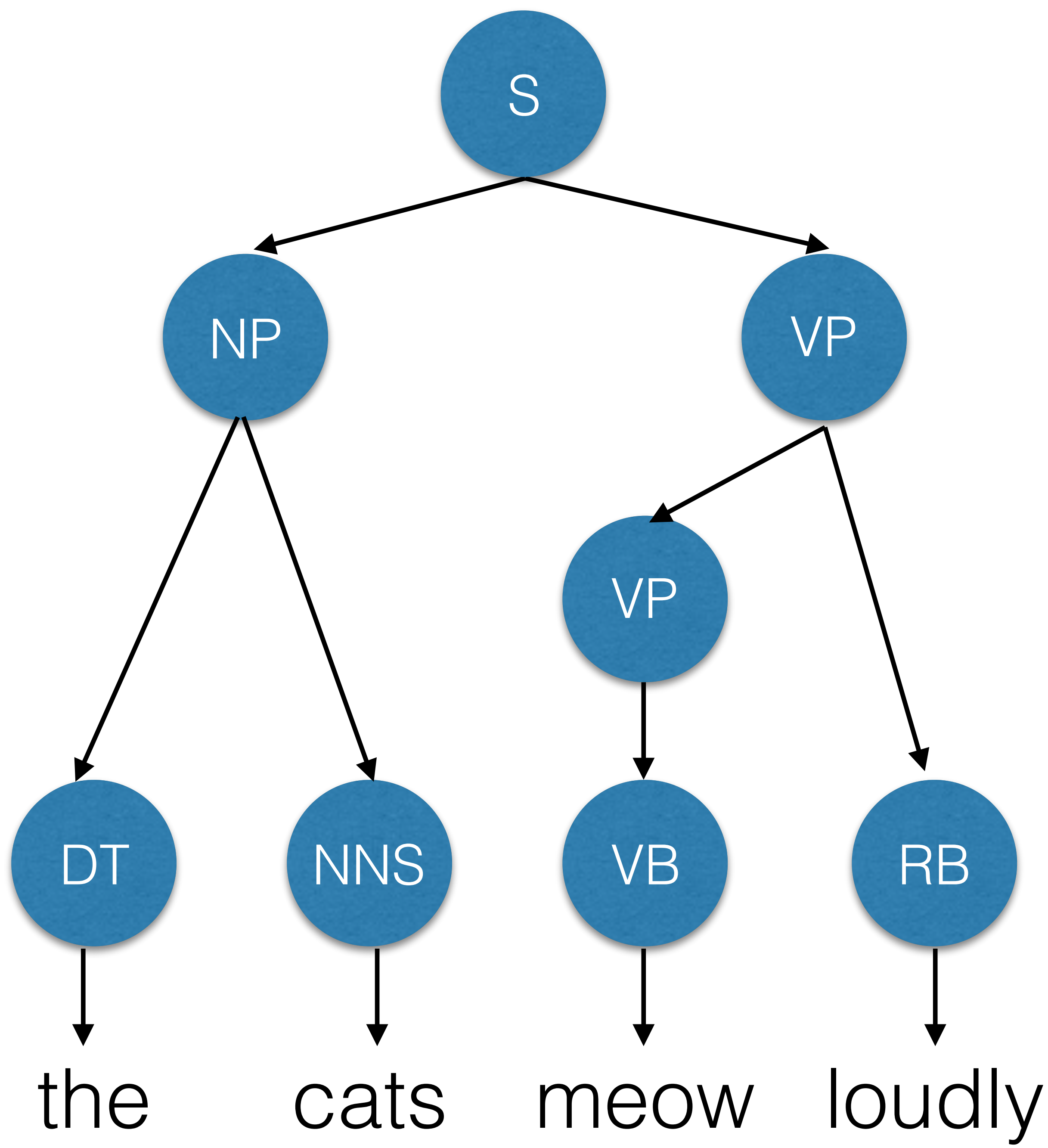


$$\pi(y) = \text{parent of node } y$$

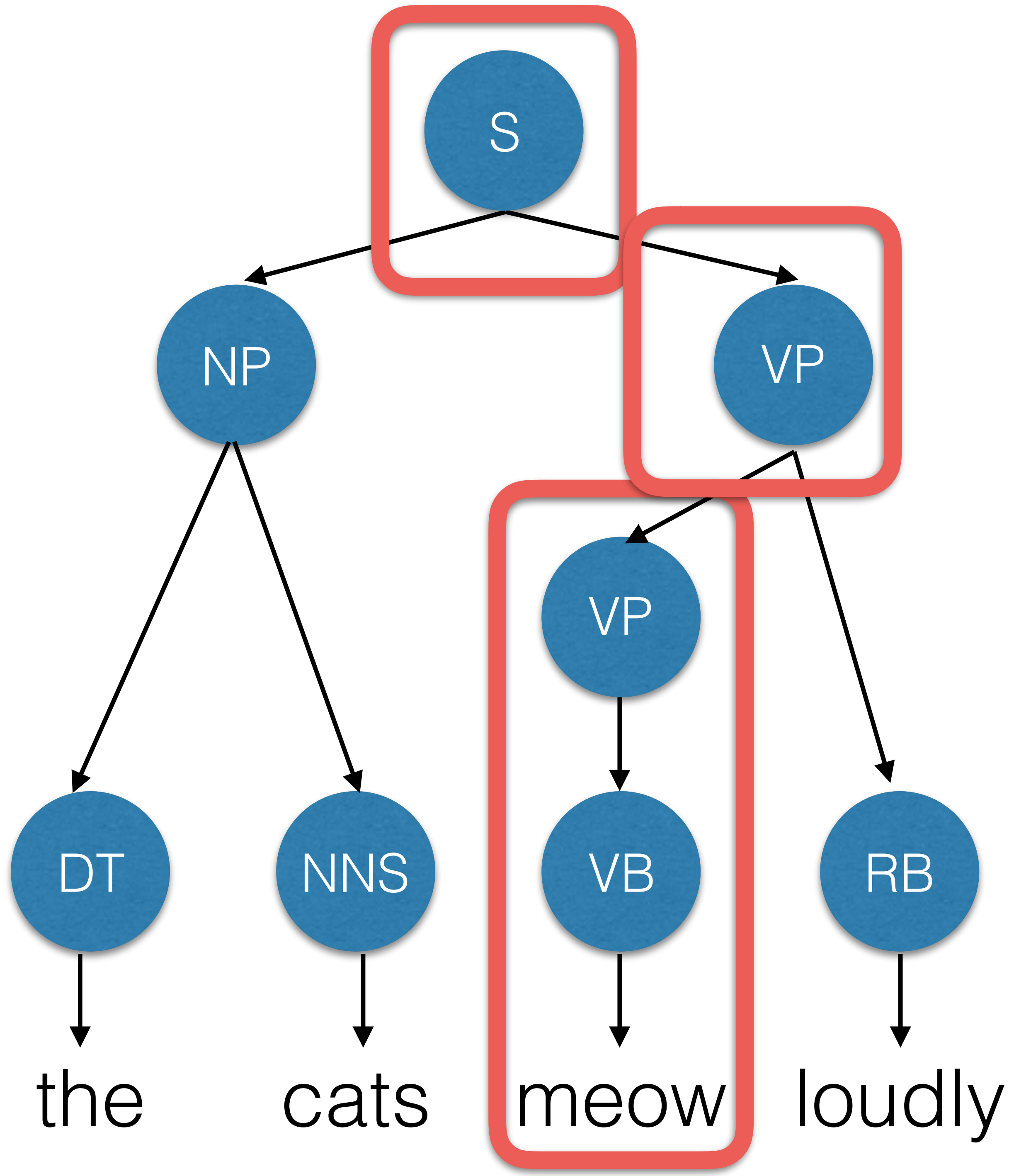
$$\mathbf{h}_y = \tanh\left(\mathbf{W}\mathbf{h}_{\pi(y)} + \mathbf{b}\right)$$

Top-Down Tree RNN

- By changing the initial state, we can build an encoder-decoder architecture on trees
- Intuitively, the initial vector “encodes” everything you want to generate.
- But- is this enough??



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$
$$p(\mathit{meow} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP}, \mathrm{VB})$$
$$p(\mathit{loudly} \mid \mathrm{S}, \mathrm{VP}, \mathrm{RB})$$



$p(\text{S})$

$p(\text{NP VP} \mid \text{S})$

$p(\text{DT NN} \mid \text{S}, \text{NP})$

$p(\textit{the} \mid \text{S}, \text{NP}, \text{DT})$

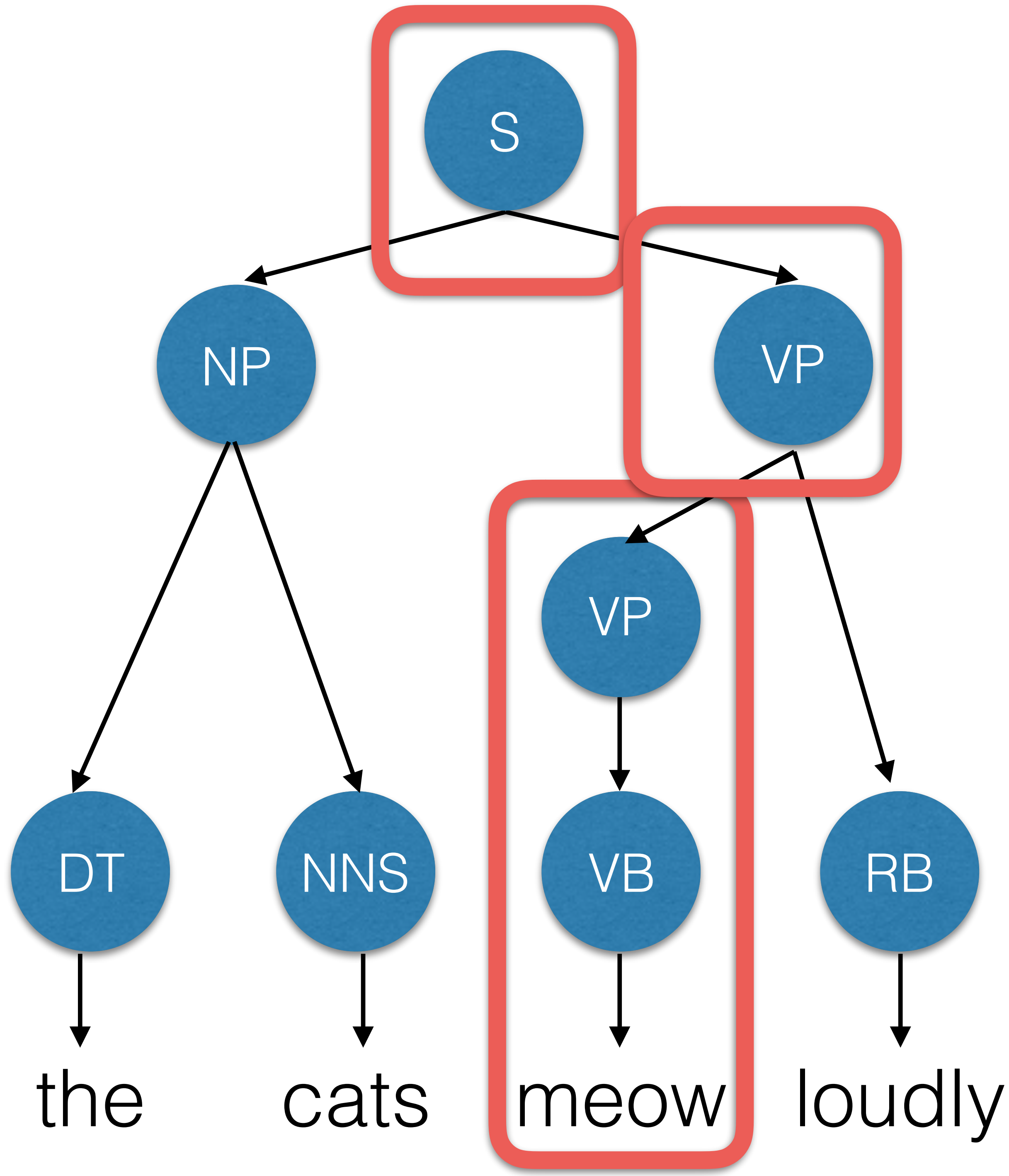
$p(\textit{cats} \mid \text{S}, \text{NP}, \text{NN})$

$p(\text{VP RB} \mid \text{S}, \text{VP})$

$p(\text{VB} \mid \text{S}, \text{VP}, \text{VP})$

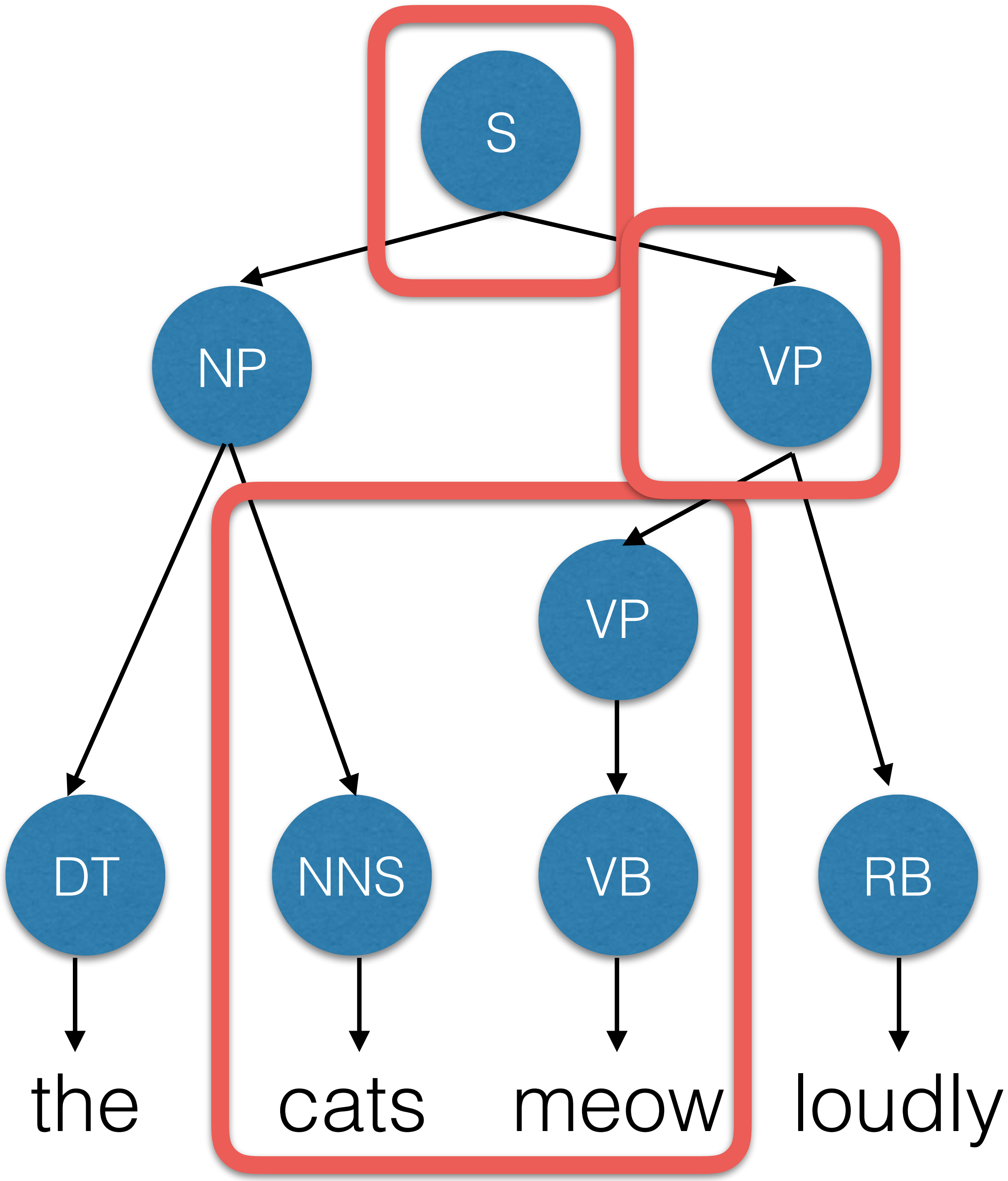
$\boxed{p(\textit{meow} \mid \text{S}, \text{VP}, \text{VP}, \text{VB})}$

$p(\textit{loudly} \mid \text{S}, \text{VP}, \text{RB})$



$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$
$$\boxed{p(\mathit{meow} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP}, \mathrm{VB})}$$
$$p(\mathit{loudly} \mid \mathrm{S}, \mathrm{VP}, \mathrm{RB})$$

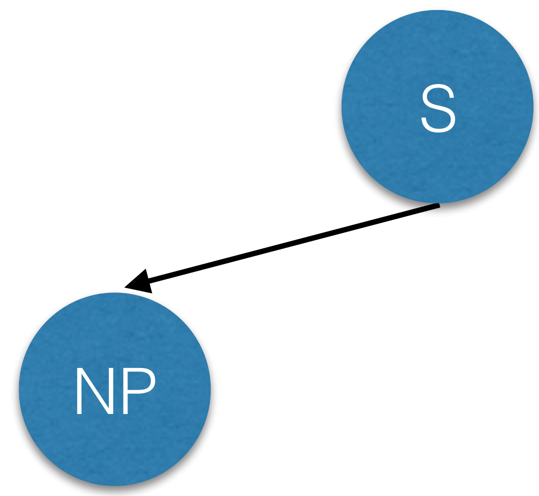
Problem: model doesn’t condition on the noun decision! Agreement??

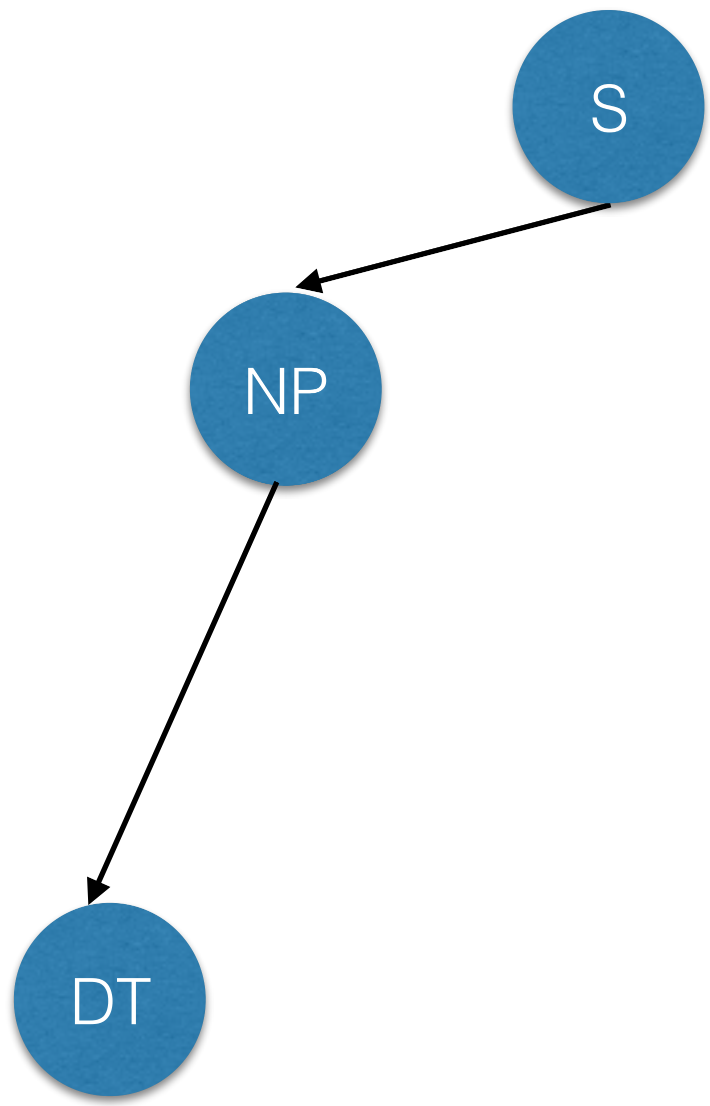


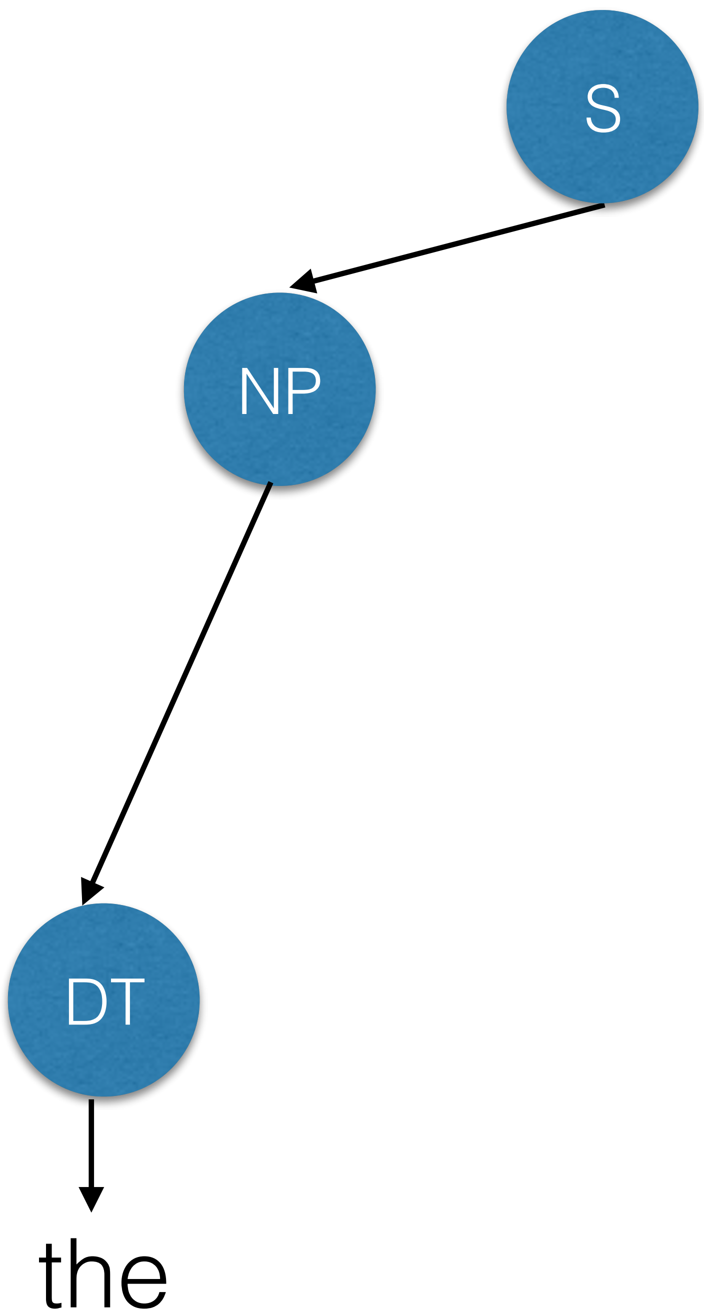
$$p(\mathrm{S})$$
$$p(\mathrm{NP\ VP} \mid \mathrm{S})$$
$$p(\mathrm{DT\ NN} \mid \mathrm{S}, \mathrm{NP})$$
$$p(\mathit{the} \mid \mathrm{S}, \mathrm{NP}, \mathrm{DT})$$
$$p(\mathit{cats} \mid \mathrm{S}, \mathrm{NP}, \mathrm{NN})$$
$$p(\mathrm{VP\ RB} \mid \mathrm{S}, \mathrm{VP})$$
$$p(\mathrm{VB} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP})$$
$$\boxed{p(\mathit{meow} \mid \mathrm{S}, \mathrm{VP}, \mathrm{VP}, \mathrm{VB})}$$
$$p(\mathit{loudly} \mid \mathrm{S}, \mathrm{VP}, \mathrm{RB})$$

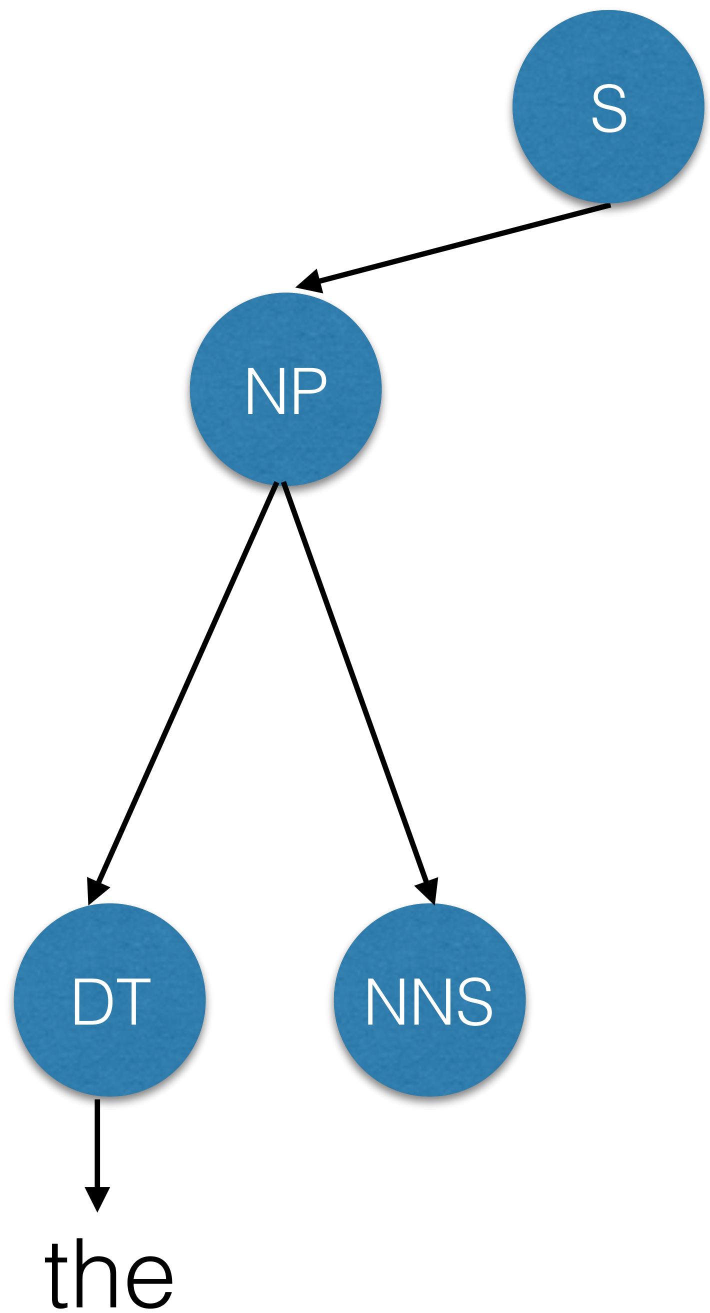
Problem: model doesn't condition on the noun decision! Agreement??

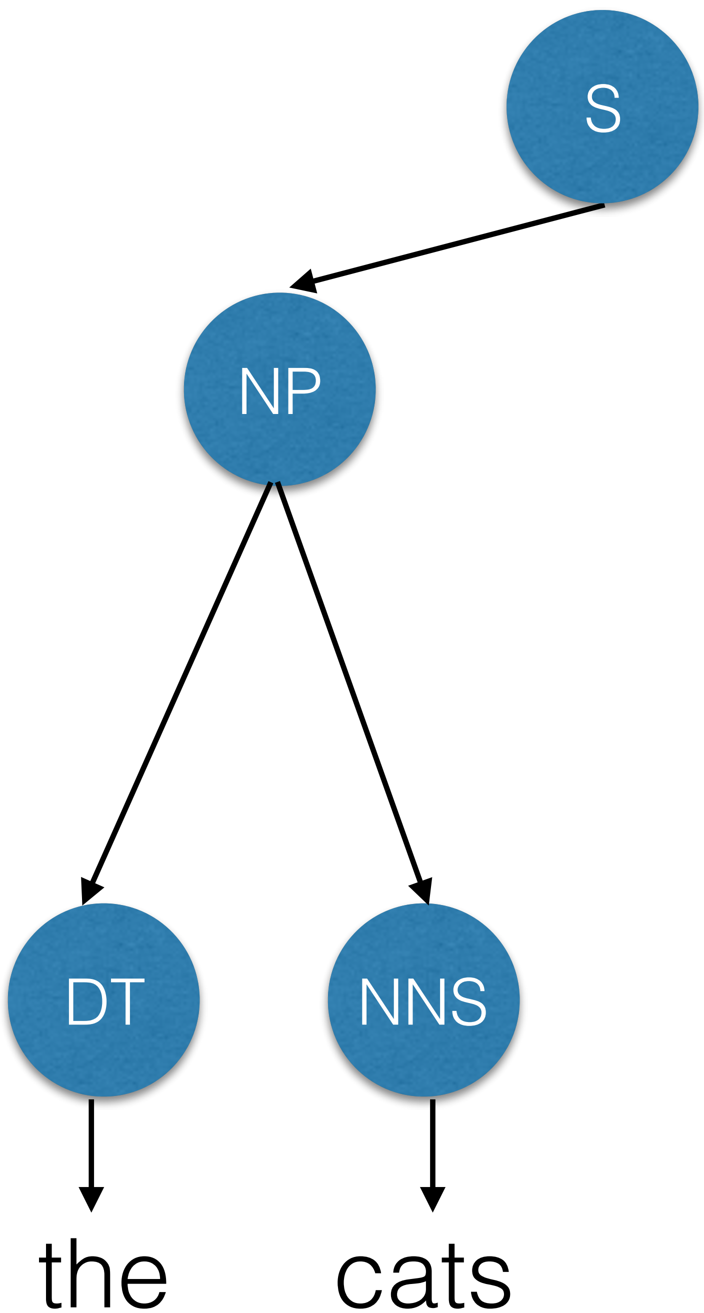

S

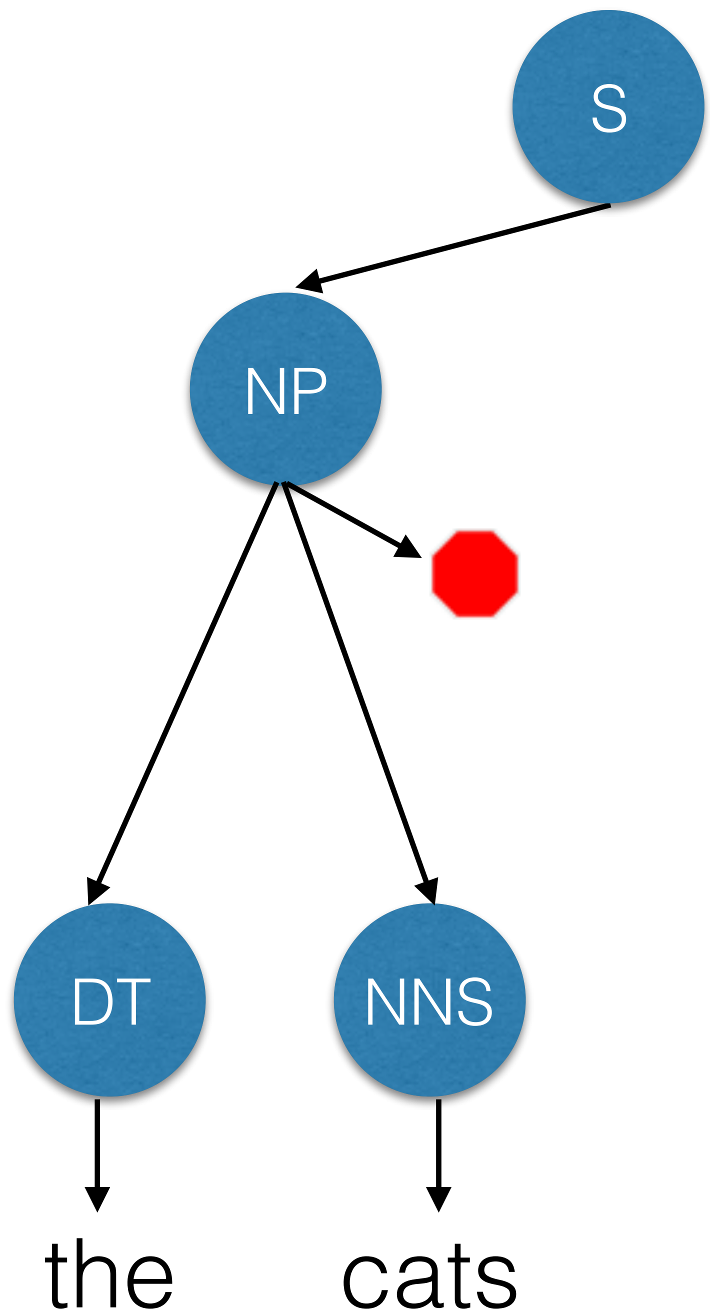

S
NP

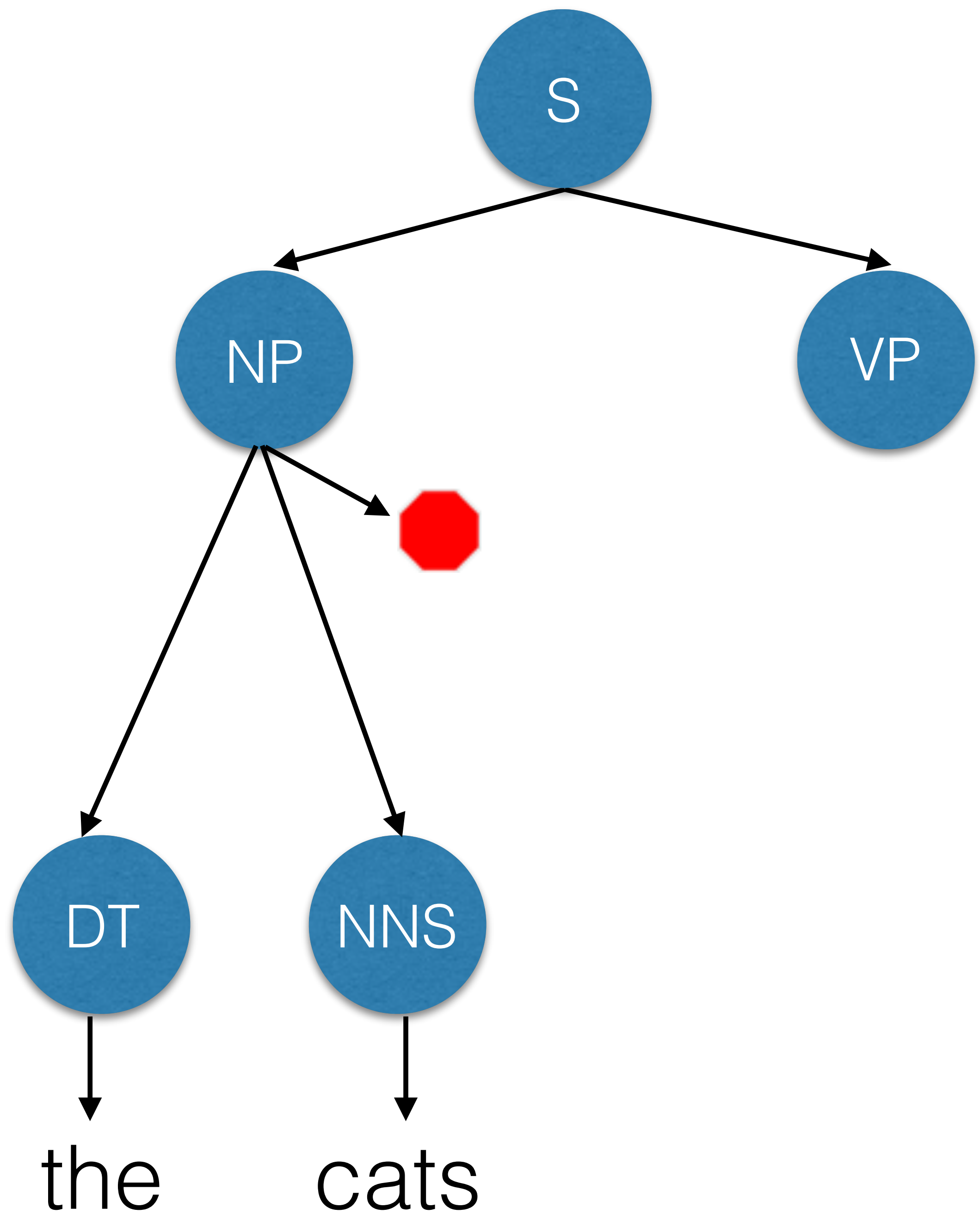

S
NP
DT

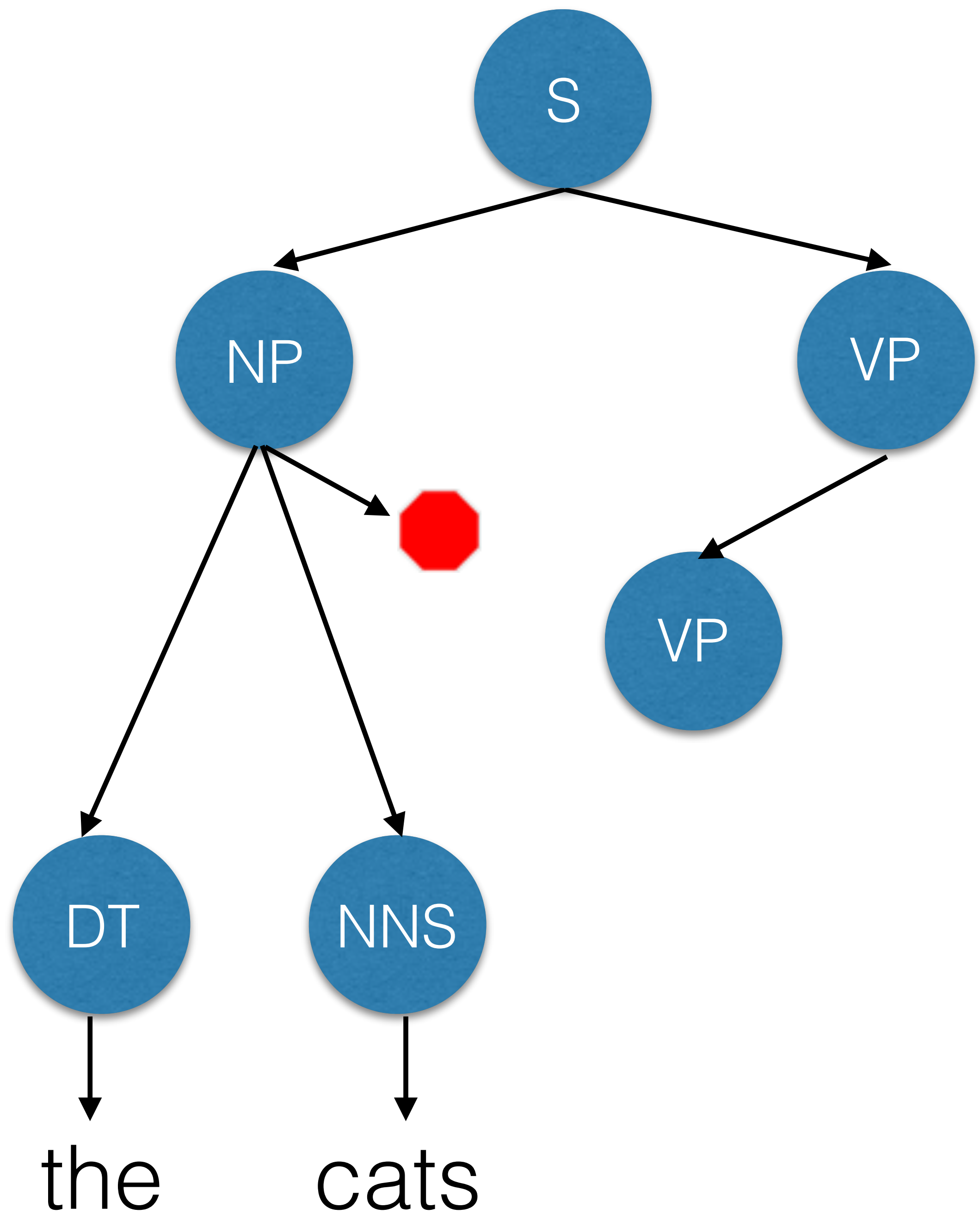

S
NP
DT
the

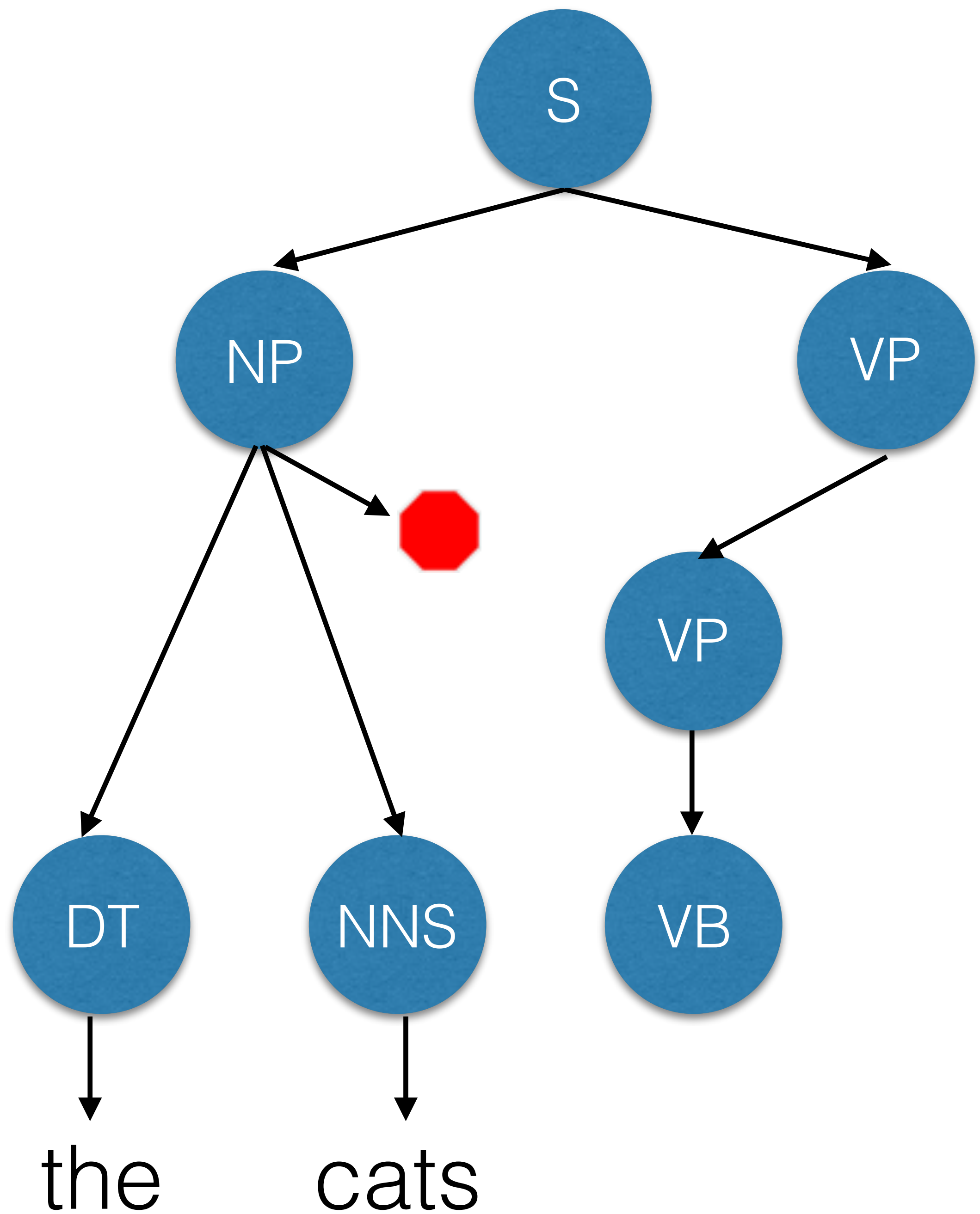

S
NP
DT
NNS
the

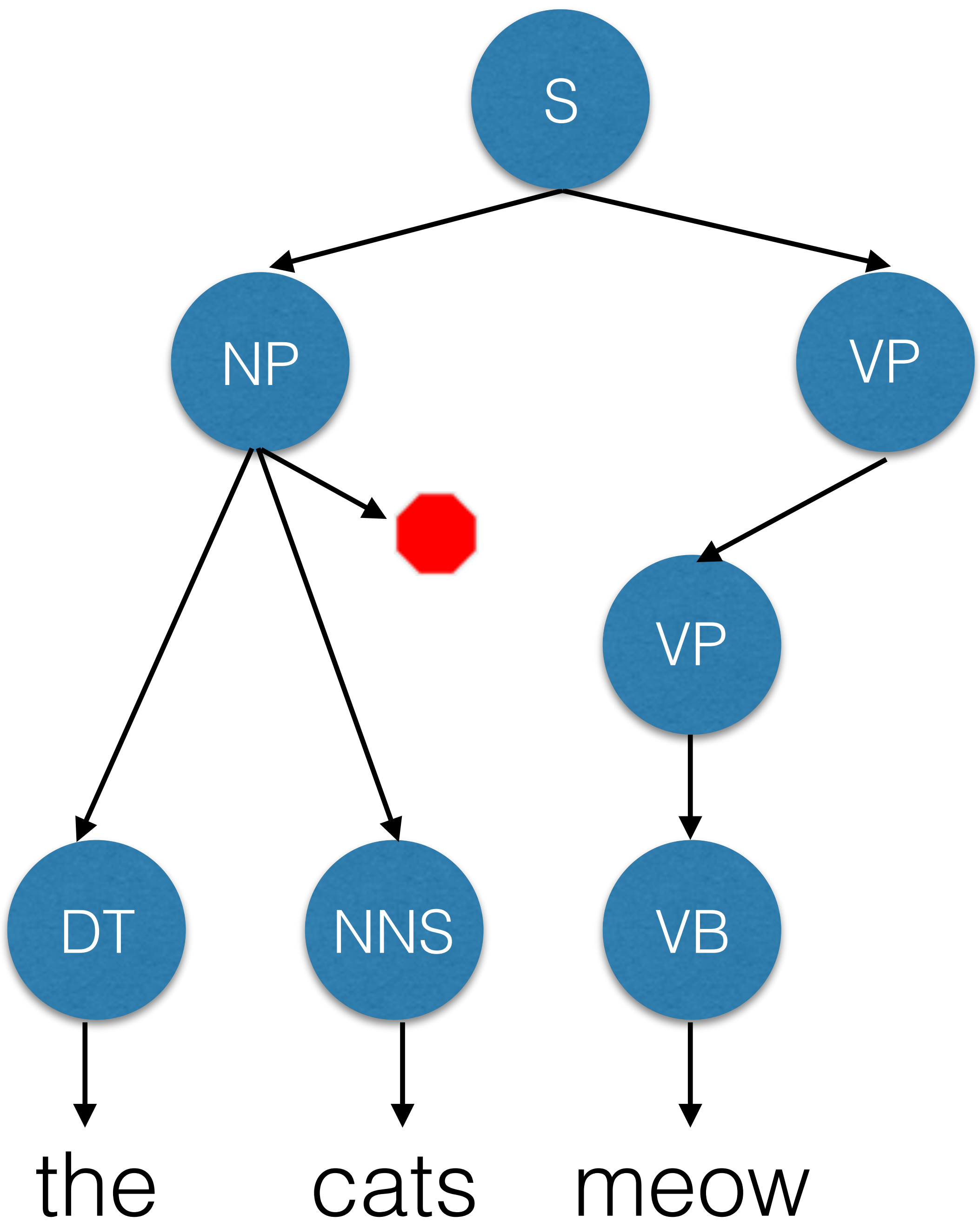

S
NP
DT
NNS
the
cats

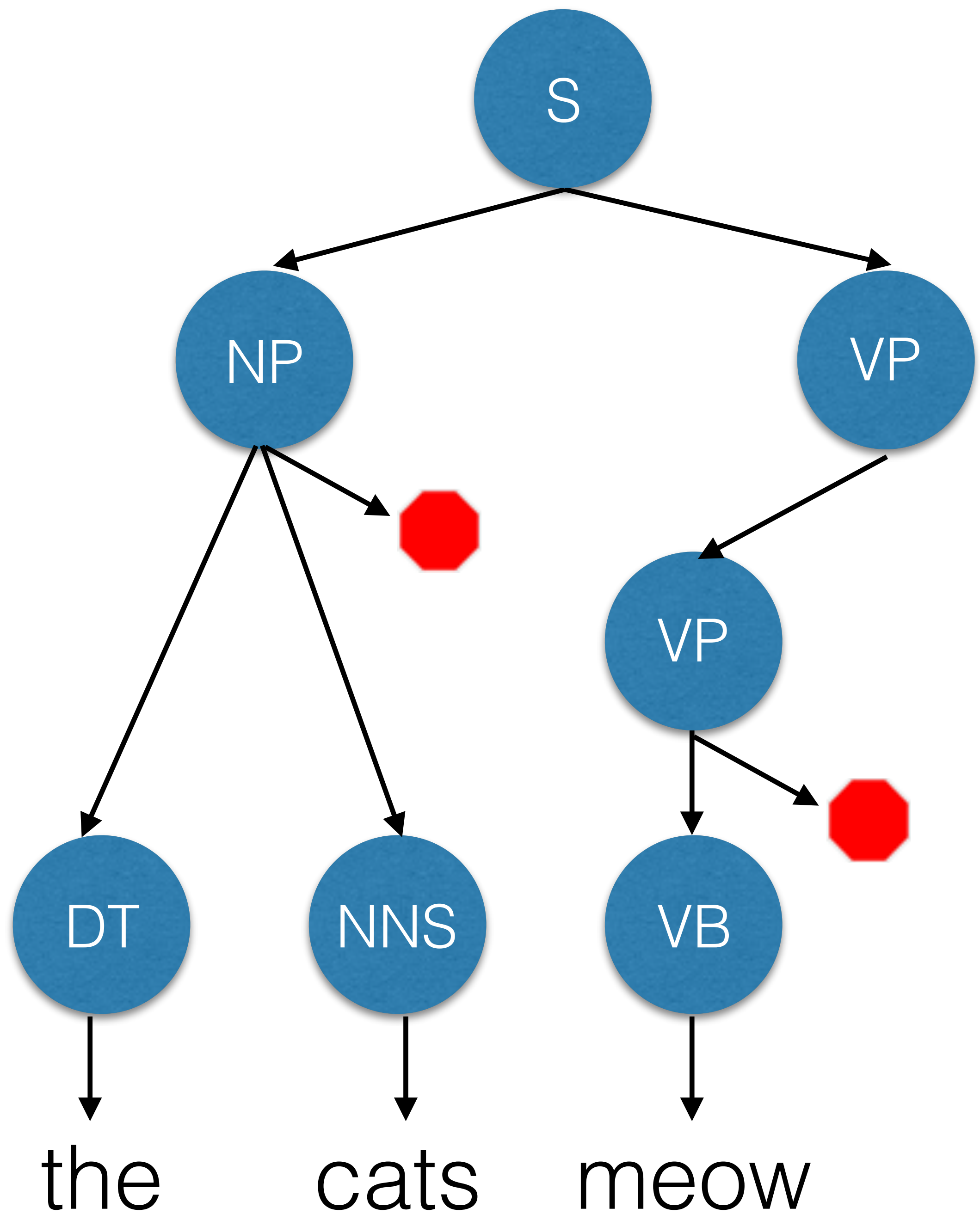

S
NP
DT
NNS
the
cats

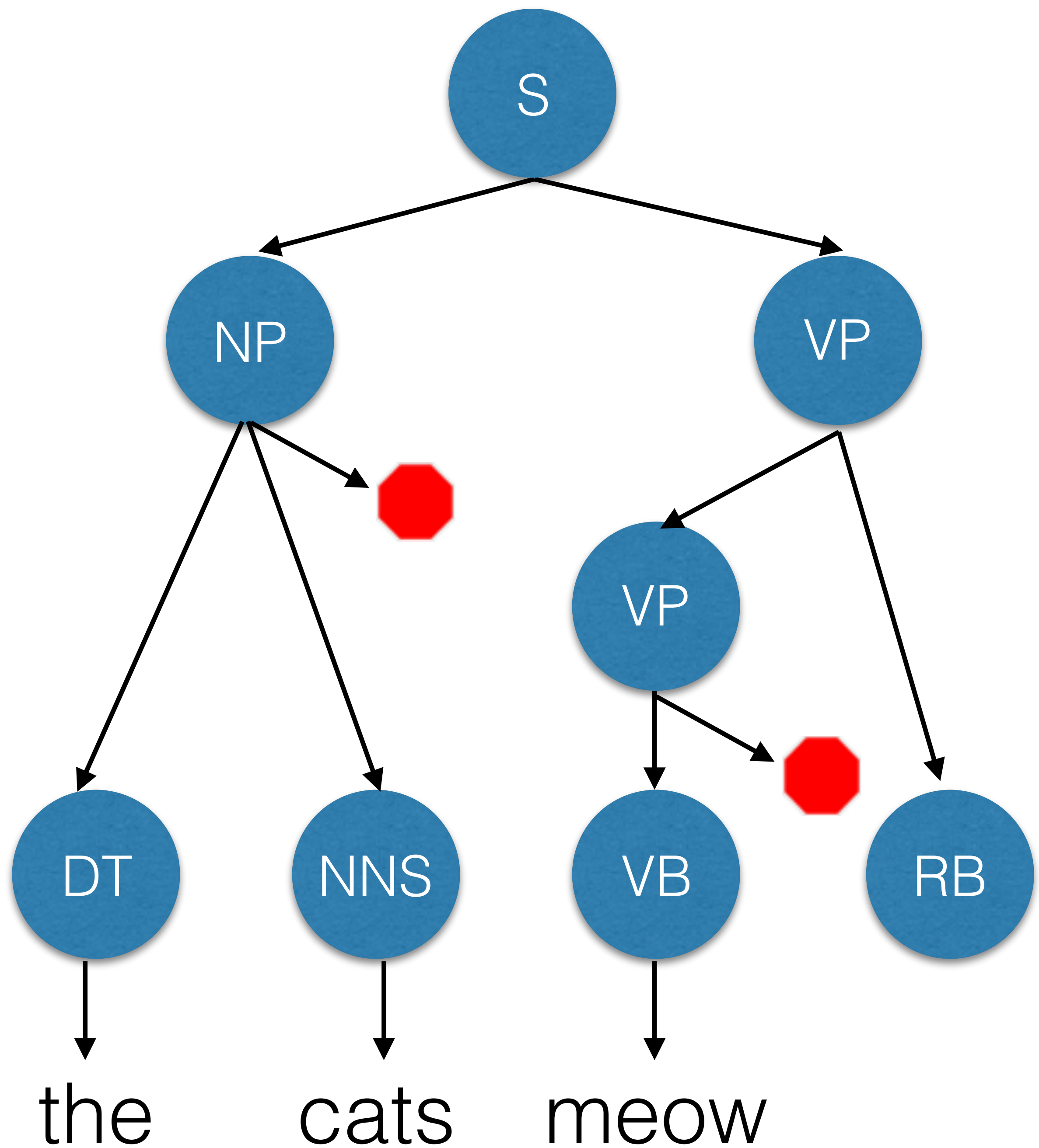

S
NP
VP
DT
NNS
the
cats

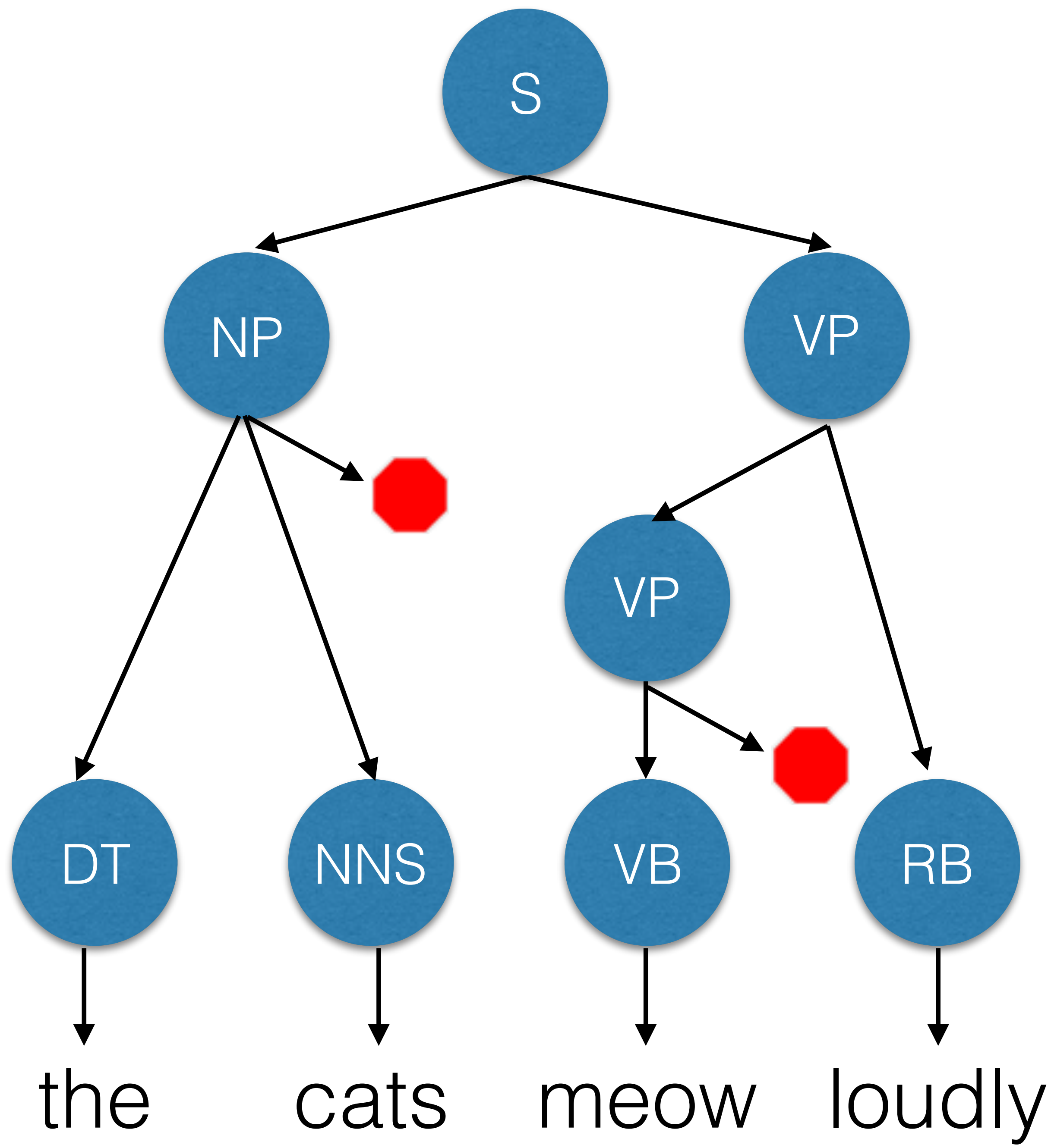

S
NP
VP
VP
DT
NNS
the
cats

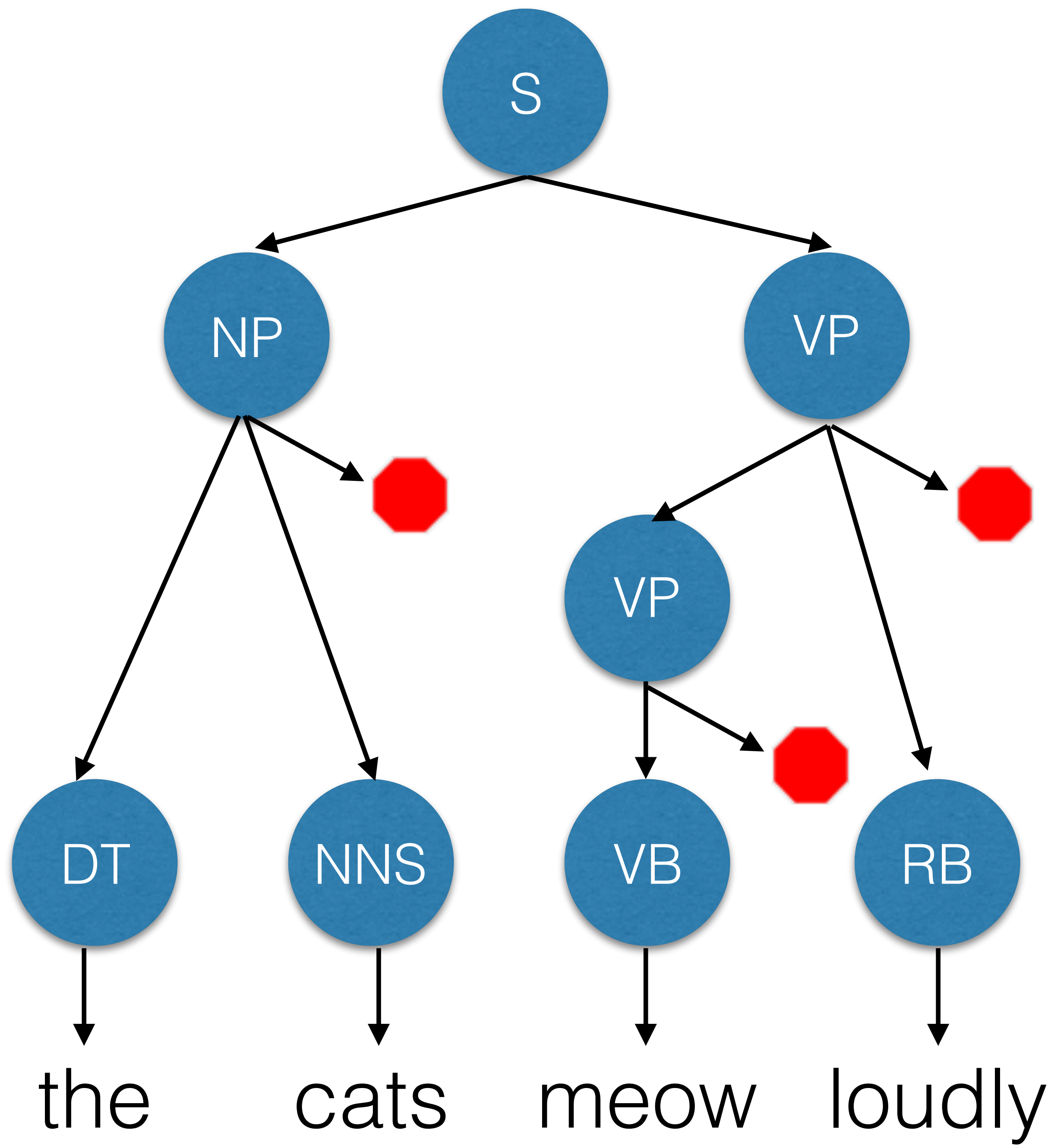

S
NP
VP
VP
DT
NNS
VB
the
cats

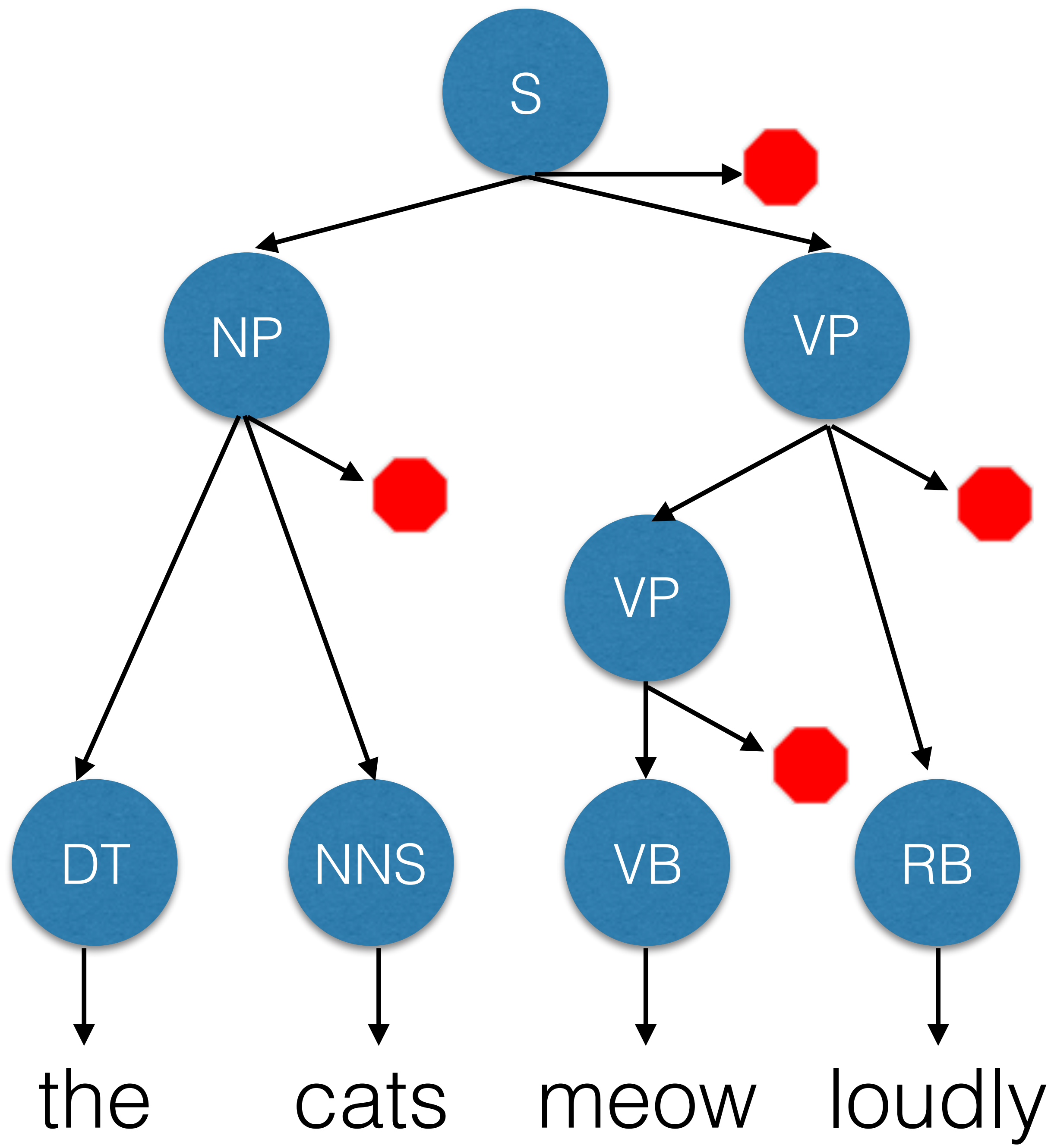

S
NP
VP
VP
DT
NNS
VB
the
cats
meow


S
NP
VP
VP
DT
NNS
VB
the
cats
meow


S
NP
VP
VP
DT
NNS
VB
RB
the
cats
meow

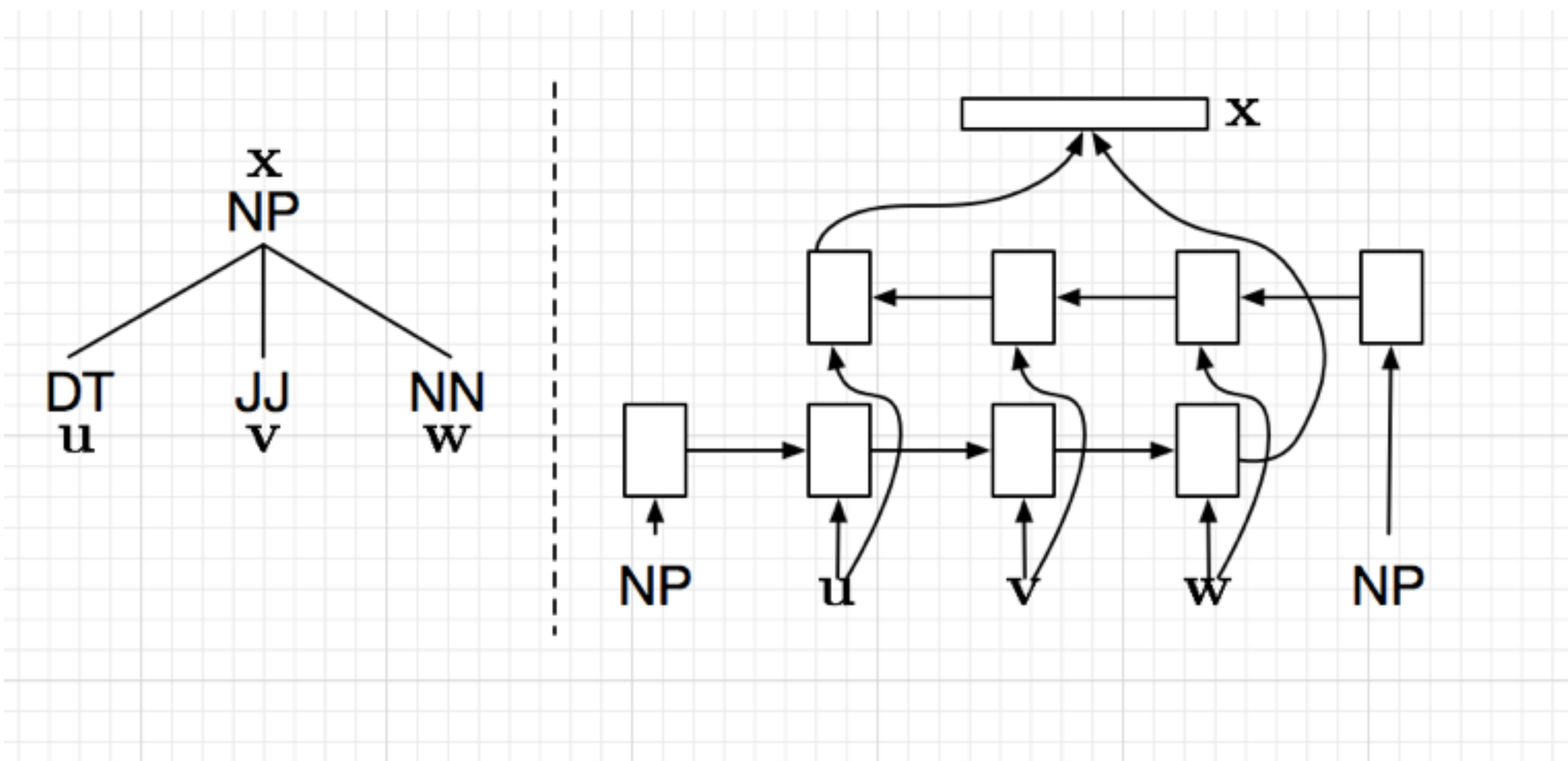

S
NP
VP
VP
DT
NNS
VB
RB
the
cats
meow
loudly


S
NP
VP
VP
DT
NNS
VB
RB
the
cats
meow
loudly


S
NP
VP
VP
DT
NNS
VB
RB
the
cats
meow
loudly

Stack	**Action**
	nt(S)
(S	nt(NP)
(S (NP	nt(DT)
(S (NP (DT	shift
(S (NP (DT the)	nt(NNS)
(S (NP (DT the) (NNS	shift
(S (NP (DT the) (NNS cats)	reduce
(S (NP (DT the) (NNS cats))	nt(VP)
(S (NP (DT the) (NNS cats)) (VP	nt(VP)
(S (NP (DT the) (NNS cats)) (VP (VP	nt(VB)
(S (NP (DT the) (NNS cats)) (VP (VP (VB	shift
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow)	reduce
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow))	nt(RB)
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow)) (RB	shift
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow)) (RB loudly)	reduce
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow)) (RB loudly))	reduce
(S (NP (DT the) (NNS cats)) (VP (VP (VB meow)) (RB loudly)))	

Composition Functions



Top-down transition-based parsing

- Can be used for both generation and parsing

Other Neural Architectures

- Hidden RNNs?

Hidden RNNs

Replace the Markov model in an HMM with an RNN.

$$\begin{aligned} y_0 &= \text{START} \\ y_i \mid \boldsymbol{y}_{<i} &\sim \text{RNNLM}(\boldsymbol{y}_{<i}) \\ x_i \mid y_i &\sim \text{Categorical}(\theta_{y_i}) \end{aligned}$$

Is this a valid model? **Yes!**

Can you perform supervised training? **Yes, easily!**

Can you perform posterior inference on $\boldsymbol{y} \mid \boldsymbol{x}$?

Well … the naive algorithm works. What about Viterbi?

Summary

- Neural Networks are expressive
  - …but structured prediction is too!
- Hybrid architectures give us the best of both worlds.