CS11-747 Neural Networks for NLP

Neural Semantic Parsing

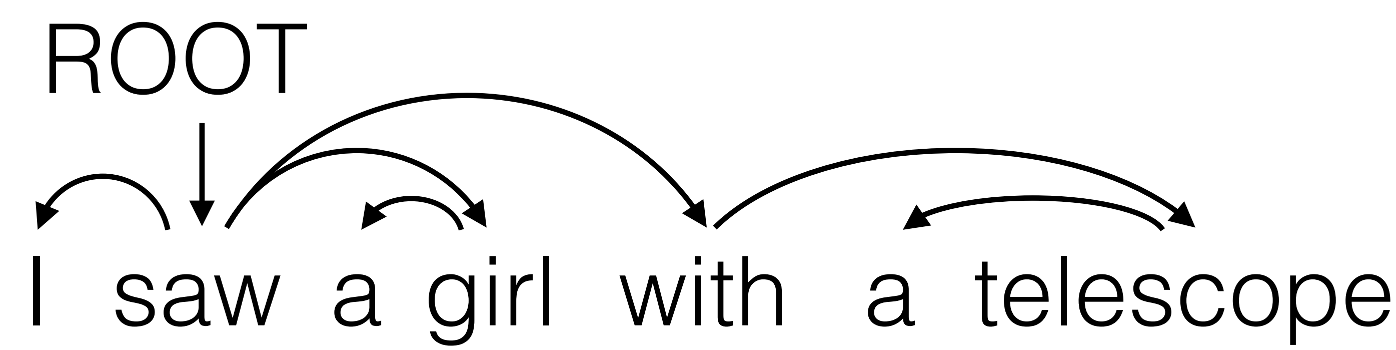
Graham Neubig

Carnegie Mellon University
Language Technologies Institute

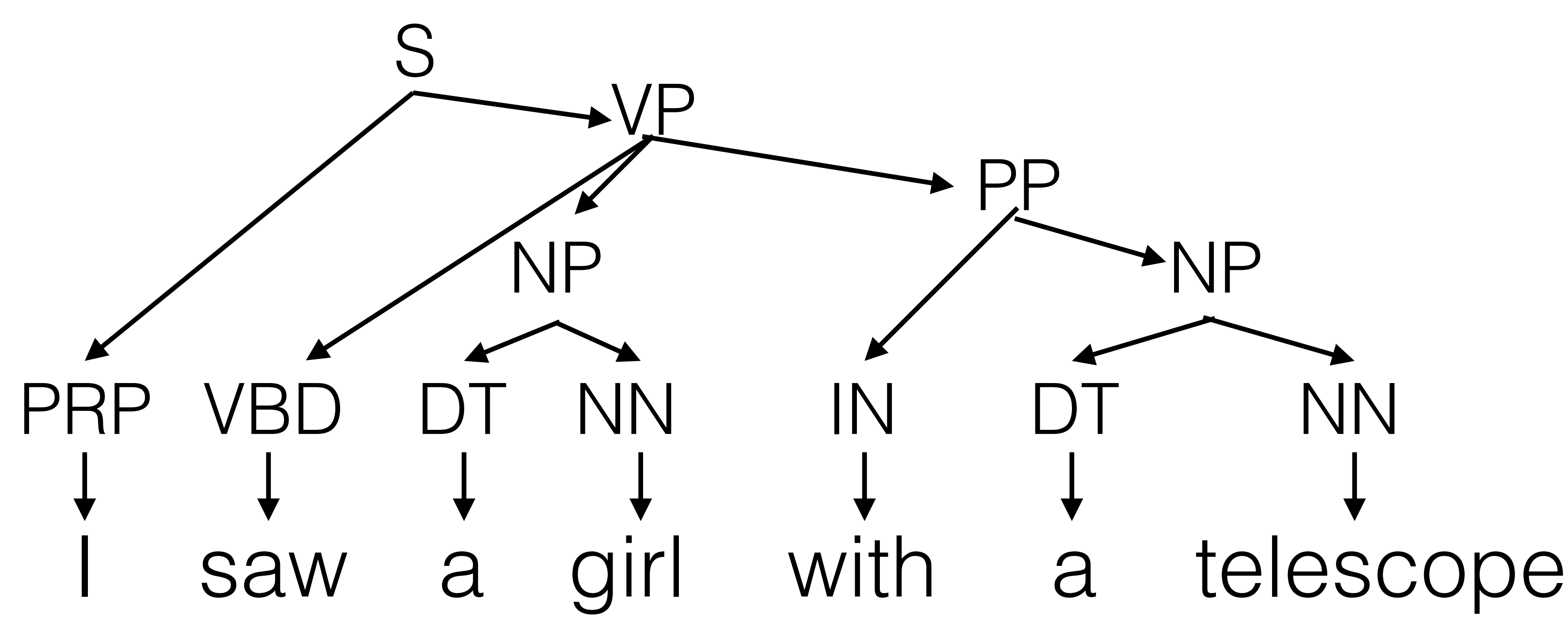
Site
https://phontron.com/class/nn4nlp2018/

Tree Structures of Syntax

- **Dependency:** focus on relations between words



- **Phrase structure:** focus on the structure of the sentence



Representations of Semantics

- Syntax only gives us the sentence structure
- We would like to know what the **sentence really means**
- Specifically, in an **grounded and operationalizable way**, so a machine can
  - Answer questions
  - Follow commands
  - etc.

Meaning Representations

- **Special-purpose representations:** designed for a specific task

- **General-purpose representations:** designed to be useful for just about anything

- **Shallow representations:** designed to only capture part of the meaning (for expediency)

Parsing to Special-purpose Meaning Representations

Example Special-purpose Representations

- A database query language for sentence understanding
- A robot command and control language
- Source code in a language such as Python (?)

Example Query Tasks

- **Geoquery:** Parsing to Prolog queries over small database (Zelle and Mooney 1996)

x: *"what is the population of iowa ?"*

y:
```
_answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```

- **Free917:** Parsing to Freebase query language (Cai and Yates 2013)

1. What are the neighborhoods in New York City?
   $\lambda x \,.\, \mathtt{neighborhoods}(\mathtt{new\_york}, x)$
2. How many countries use the rupee?
   $\mathtt{count}(x) \,.\, \mathtt{countries\_used}(\mathtt{rupee}, x)$

- Many others: WebQuestions, WikiTables, etc.

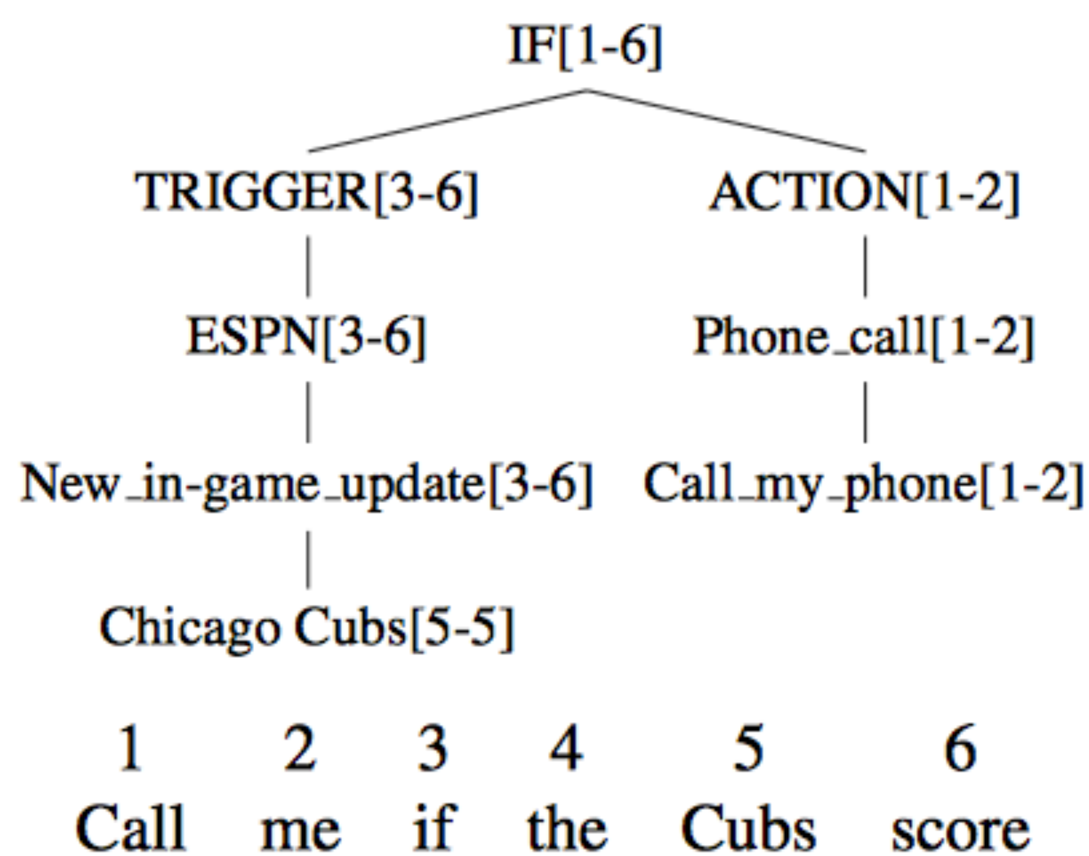
Example Command and Control Tasks

- **Robocup**: Robot command and control (Wong and Mooney 2006)

```
((bowner our {4})
 (do our {6} (pos (left (half our)))))
```

If our player 4 has the ball, then our player 6 should stay in the left side of our half.

- **If this then that:** Commands to smartphone interfaces (Quirk et al. 2015)



Example Code Generation Tasks

- Hearthstone cards (Ling et al. 2015)


```
class DivineFavor(SpellCard):
    def __init__(self):
        super().__init__("Divine Favor", 3,
    CHARACTER_CLASS.PALADIN, CARD_RARITY.RARE)

    def use(self, player, game):
        super().use(player, game)
        difference = len(game.other_player.hand)
      - len(player.hand)
        for i in range(0, difference):
            player.draw()
```

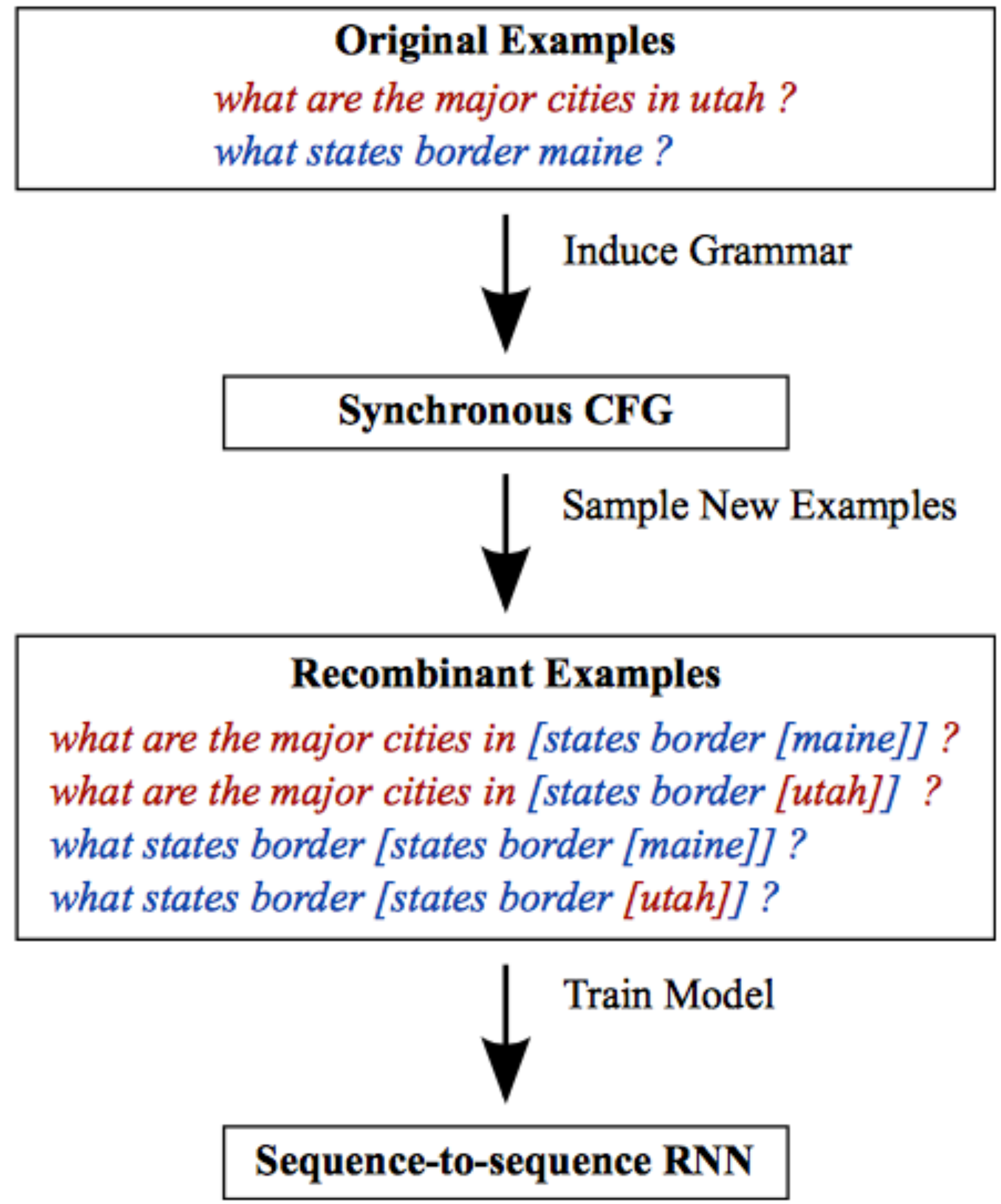
- Django commands (Oda et al. 2015)

convert cull_frequency into an integer and substitute it for self._cull_frequency.

```
self._cull_frequency = int(cull_frequency)
```

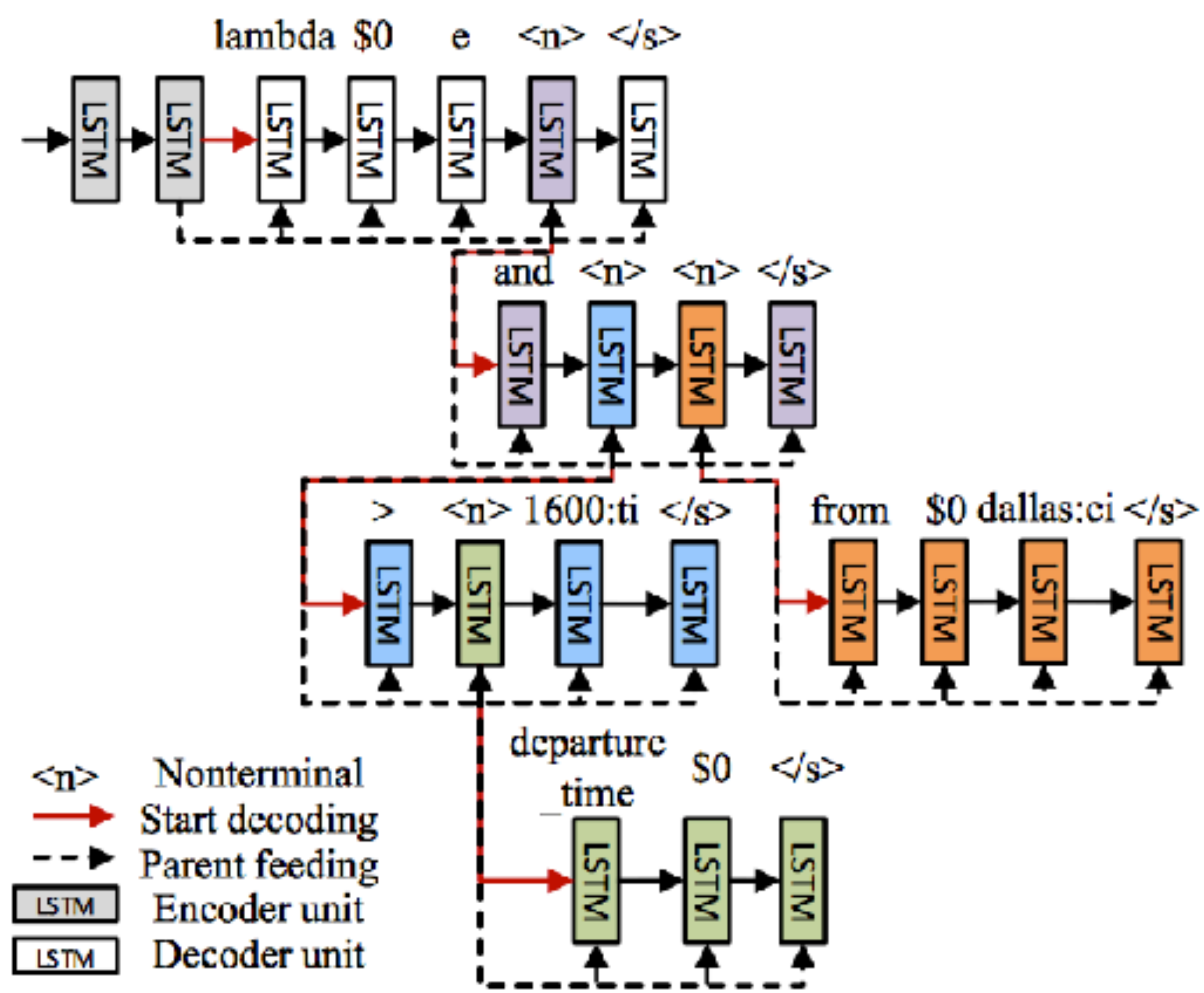
A First Attempt: Sequence-to-sequence Models (Jia and Liang 2016)

- Simple string-based sequence-to-sequence model
- Doesn't work well as-is, so generate extra synthetic data from a CFG



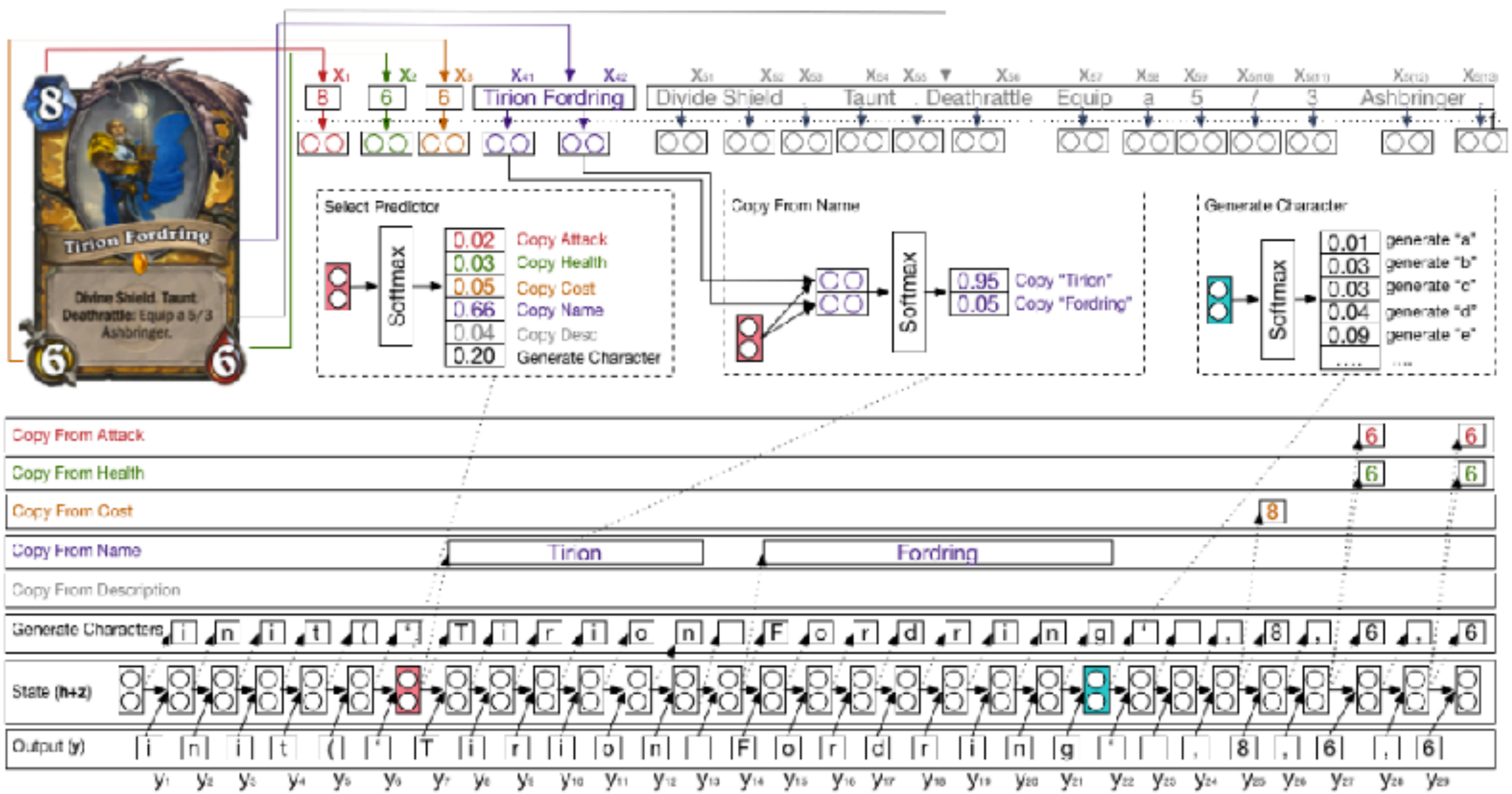
A Better Attempt: Tree-based Parsing Models

- Generate from top-down using hierarchical sequence-to-sequence model (Dong and Lapata 2016)



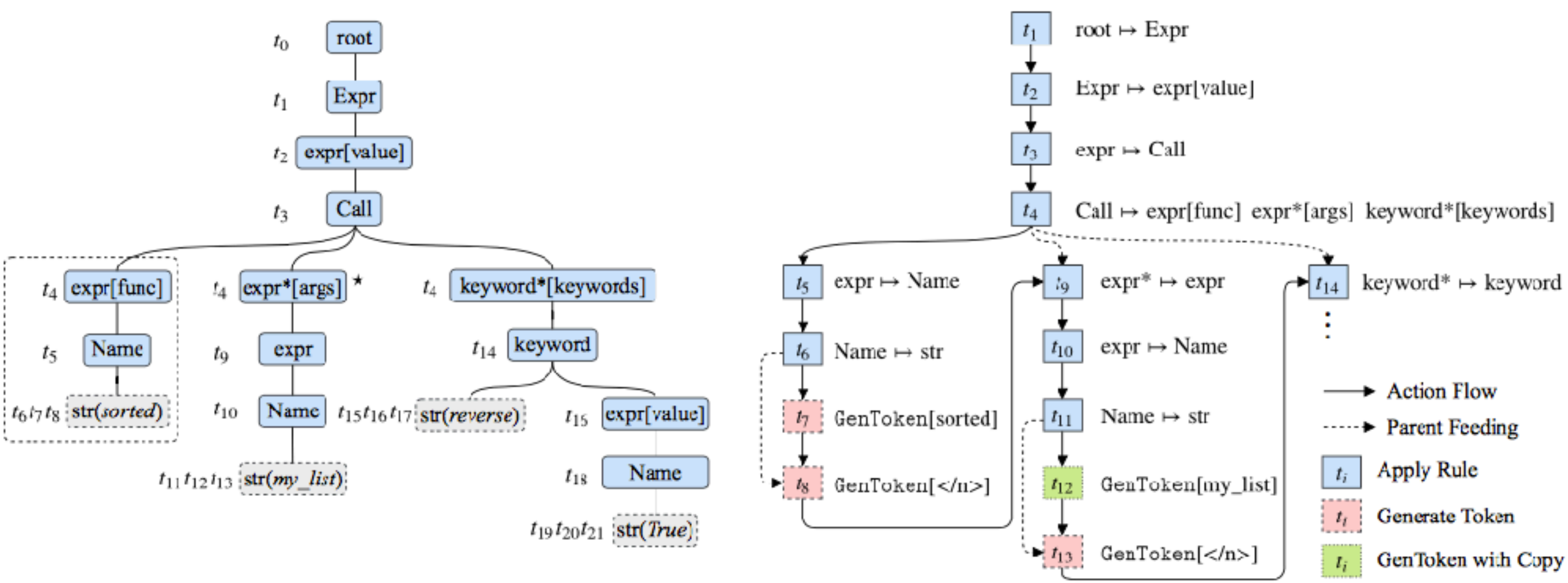
Code Generation: Character-based Generation+Copy

- In source code (or other semantic parsing tasks) there is a significant amount of copying
- **Solution:** character-based generation+copy, w/ clever independence assumptions to make training easy (Ling et al. 2016)



Code Generation: Handling Syntax

- Code also has syntax, e.g. in form of Abstract Syntax Trees (ASTs)
- Tree-based model that generates AST obeying code structure and using to modulate information flow (Yin and Neubig 2017)



(a) (b)

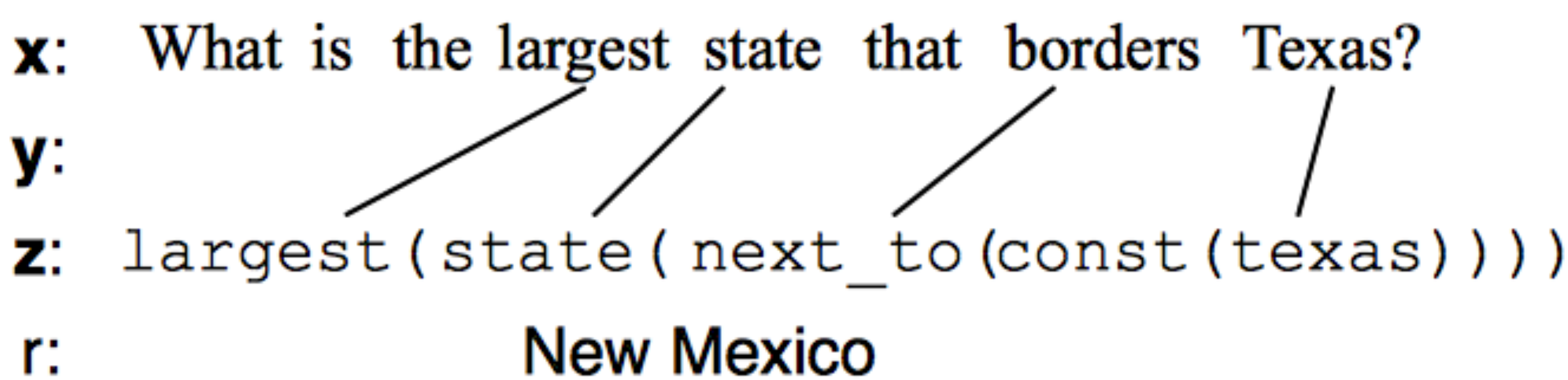
Input: `sort my_list in descending order`

Code: `sorted(my_list, reverse=True)`

Learning Signals for Semantic Parsing

Supervised Learning

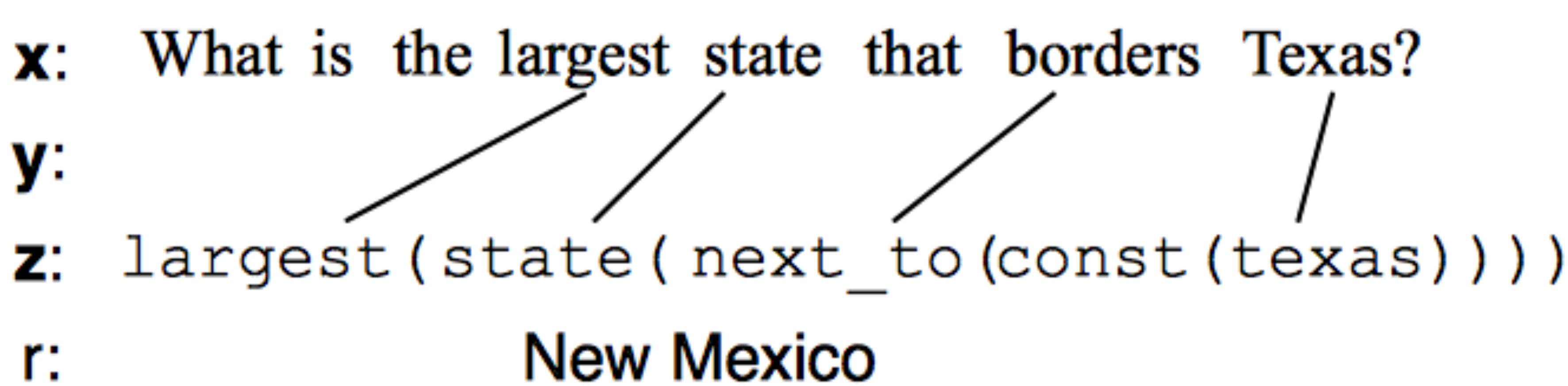
- For a natural language utterance, manually annotate its representation



- Standard datasets:
    - GeoQuery (questions about US Geography)
    - ATIS (flight booking)
    - RoboCup (robot command and control)
- Problem: costly to create!

Weakly Supervised Learning

- Sometimes we don't have annotated logical forms
- Treat logical forms as a latent variable, give a boost when we get the answer correct (Clarke et al 2010)

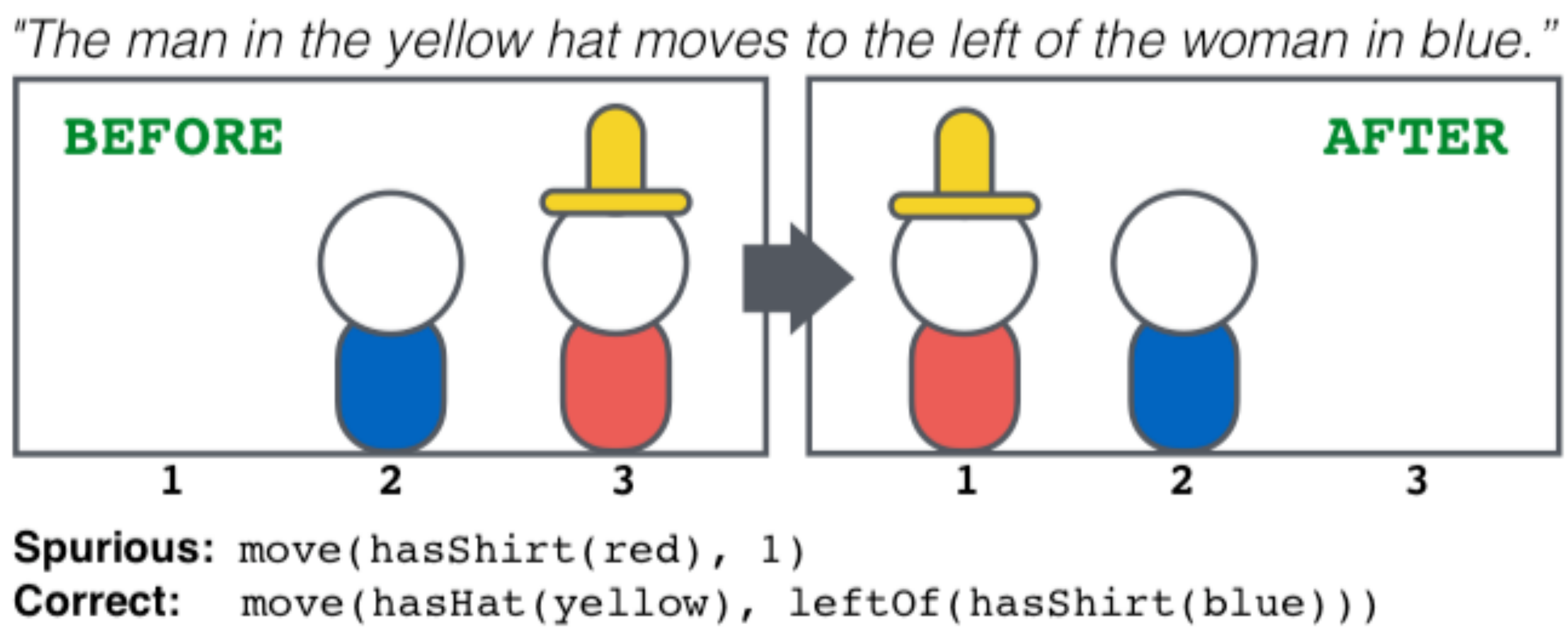


Latent

- Can be framed as a reinforcement learning problem

Problem w/ Weakly Supervised Learning: Spurious Logical Forms

- Sometimes you can get the right answer without actually doing the generalizable thing (Guu et al. 2017)



- Can be mitigated by encouraging diversity in updates at test time (Guu et al. 2017)

Interactive Learning of Semantic Parsers

- Good thing about explicit semantic representation: is human interpretable and can be built w/ humans
- e.g. Ask users to correct incorrect SQL queries (Iyer et al. 2017)
- e.g. Building up a "library" of commands to perform complex tasks (Wang et al. 2017)

```
def: add palm tree
    def: brown trunk height 3
        def: add brown top 3 times
            repeat 3 [add brown top]
    def: go to top of tree
        select very top of has color brown
    def: add leaves here
        def: select all sides
            select left or right or front or back
        add green
```

Parsing to General-purpose Meaning Representation

Meaning Representation Desiderata (Jurafsky and Martin 17.1)

- **Verifiability:** ability to ground w/ a knowledge base, etc.
- **Unambiguity:** one representation should have one meaning
- **Canonical form:** one meaning should have one representation
- **Inference ability:** should be able to draw conclusions
- **Expressiveness:** should be able to handle a wide variety of subject matter

First-order Logic

- Logical symbols, connective, variables, constants, etc.
  - There is a restaurant that serves Mexican food near ICSI.
    `∃xRestaurant(x)∧ Serves(x,MexicanFood)∧ Near((LocationOf(x),LocationOf(ICSI))`
  - All vegetarian restaurants serve vegetarian food.
    `∀xVegetarianRestaurant(x) ⇒ Serves(x,VegetarianFood)`
- Lambda calculus allows for expression of functions
  `λx.λy.Near(x,y)(Bacaro)`
  `λy.Near(Bacaro,y)`

Abstract Meaning Representation

(Banarescu et al. 2013)

- Designed to be simpler and easier for humans to read

- Graph format, with arguments that mean the same thing linked together

- Large annotated sembank available

LOGIC format:

$\exists$ w, b, g:
instance(w, want-01) $\wedge$ instance(g, go-01) $\wedge$
instance(b, boy) $\wedge$ arg0(w, b) $\wedge$
arg1(w, g) $\wedge$ arg0(g, b)

AMR format (based on PENMAN):

```
(w / want-01
   :arg0 (b / boy)
   :arg1 (g / go-01
            :arg0 b))
```

GRAPH format:

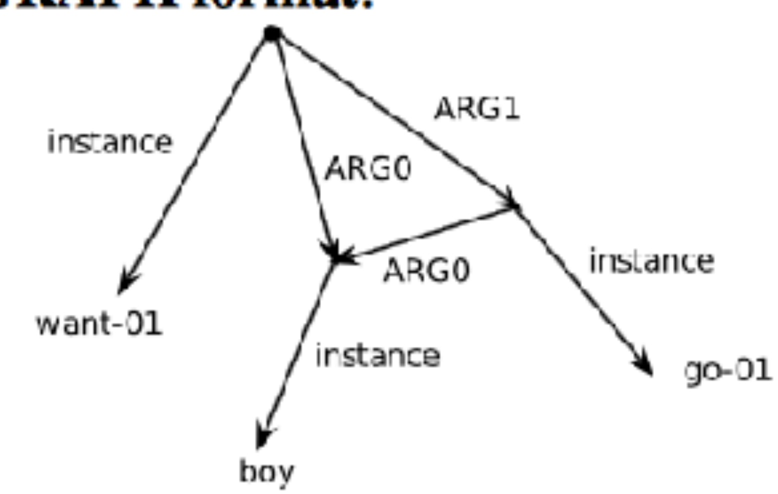
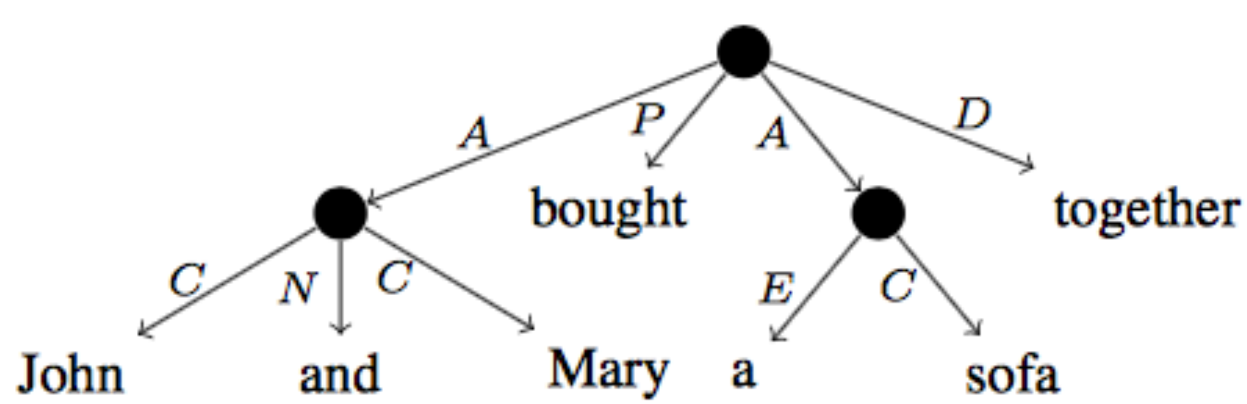


Figure 1: Equivalent formats for representating the meaning of "The boy wants to go".

Other Formalisms

- **Minimal recursion semantics** (Copestake et al. 2005): variety of first-order logic that strives to be as flat as possible to preserve ambiguity

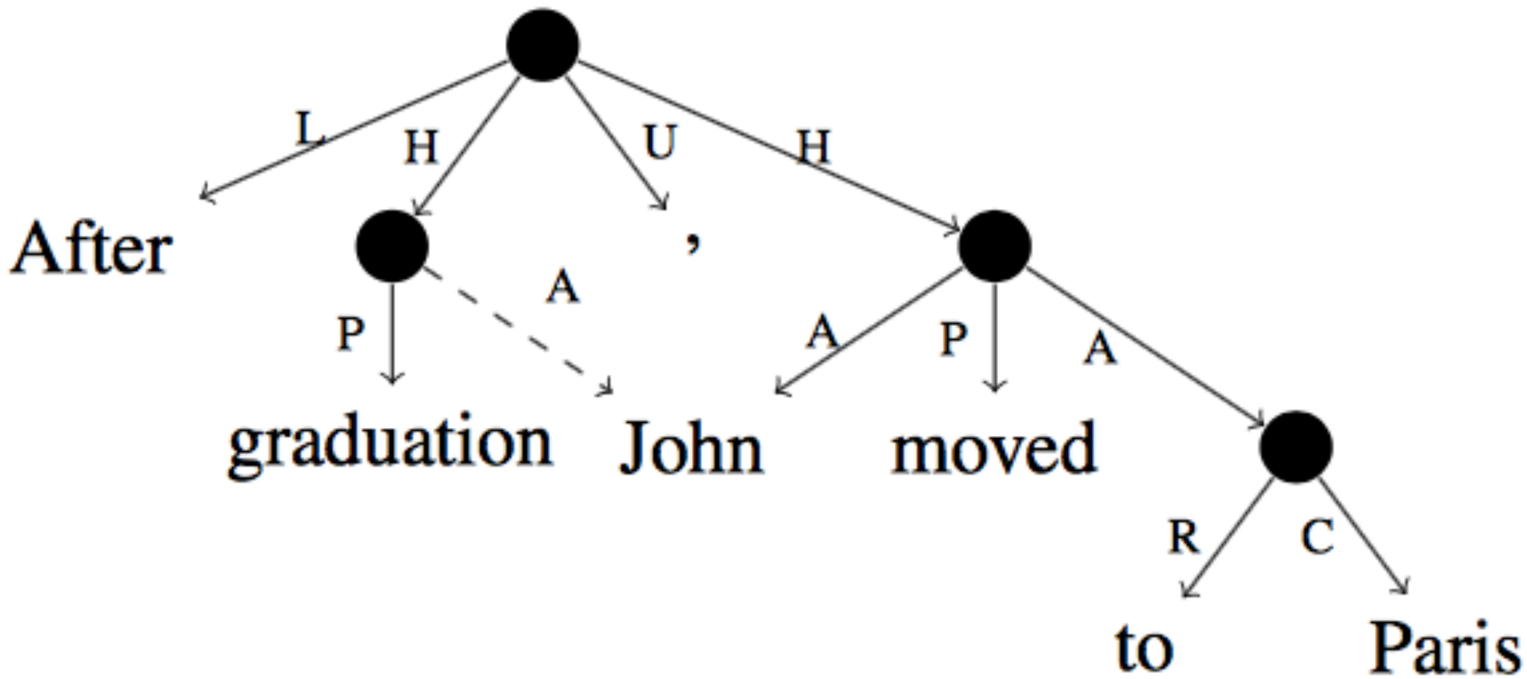
- **Universal conceptual cognitive annotation** (Abend and Rappoport 2013): Extremely course-grained annotation aiming to be universal and valid across languages



Parsing to Graph Structures

- In many semantic representations, would like to parse to directed acyclic graph
- Modify the transition system to add special actions that allow for DAGs
  - “Right arc” doesn’t reduce for AMR (Damonte et al. 2017)
  - Add “remote”, “node”, and “swap” transitions for UCCA (Hershcovich et al. 2017)
- Perform linearization and insert pseudo-tokens for re-entry actions (Buys and Blunsom 2017)

An Example (Hershcovich et al. 2017)



Before Transition				Transition	After Transition					Condition
Stack	Buffer	Nodes	Edges		Stack	Buffer	Nodes	Edges	Terminal?	
S	$x \mid B$	V	E	SHIFT	$S \mid x$	B	V	E	$-$	
$S \mid x$	B	V	E	REDUCE	S	B	V	E	$-$	
$S \mid x$	B	V	E	NODE$_X$	$S \mid x$	$y \mid B$	$V \cup \{y\}$	$E \cup \{(y,x)_X\}$	$-$	$x \neq \text{root}$
$S \mid y, x$	B	V	E	LEFT-EDGE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)_X\}$	$-$	$\begin{cases} x \notin w_{1:n}, \\ y \neq \text{root}, \\ y \not\leadsto_G x \end{cases}$
$S \mid x, y$	B	V	E	RIGHT-EDGE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)_X\}$	$-$	
$S \mid y, x$	B	V	E	LEFT-REMOTE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)_X^*\}$	$-$	
$S \mid x, y$	B	V	E	RIGHT-REMOTE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)_X^*\}$	$-$	
$S \mid x, y$	B	V	E	SWAP	$S \mid y$	$x \mid B$	V	E	$-$	$\mathrm{i}(x) < \mathrm{i}(y)$
[root]	$\emptyset$	V	E	FINISH	$\emptyset$	$\emptyset$	V	E	$+$	

Linearization for Graph Structures (Konstas et al. 2017)

- A simple method for handling trees is linearization to a sequence of symbols
- This is possible, although less easy, to do for graphs

US officials held an expert group meeting in January 2002 in New York.

```
(h / hold-04
  :ARG0 (p2 / person
    :ARG0-of (h2 / have-org-role-91
        :ARG1 (c2 / country
          :name (n3 / name
            :op1 "United" op2: "States"))
        :ARG2 (o / official)))
  :ARG1 (m / meet-03
    :ARG0 (p / person
        :ARG1-of (e / expert-01)
          :ARG2-of (g / group-01)))
  :time (d2 / date-entity :year 2002 :month 1)
  :location (c / city
    :name (n / name :op1 "New" :op2 "York")))
```

(a) US officials held an expert group meeting in January 2002 in New York.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 country :name name :op1
United :op2 States :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity :year 2002 :month 1
  :location city :name name :op1 New :op2 York
```

(b) country_0 officials held an expert group meeting in month_0 year_0 in city_1.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 country_0 :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity year_0 month_0
  :location city_1
```

(c) loc_0 officials held an expert group meeting in month_0 year_0 in loc_1.

```
hold
  :ARG0 person :ARG0-of have-org-role :ARG1 loc_0 :ARG2 official
  :ARG1 meet :ARG0 person :ARG1-of expert :ARG2-of group
  :time date-entity year_0 month_0
  :location loc_1
```

(d) loc_0 officials held an expert group meeting in month_0 year_0 in loc_1.

```
hold
  :ARG0 ( person :ARG0-of ( have-org-role :ARG1 loc_0 :ARG2 official ) )
  :ARG1 ( meet :ARG0 ( person :ARG1-of expert :ARG2-of group ) )
  :time ( date-entity year_0 month_0 )
  :location loc_1
```

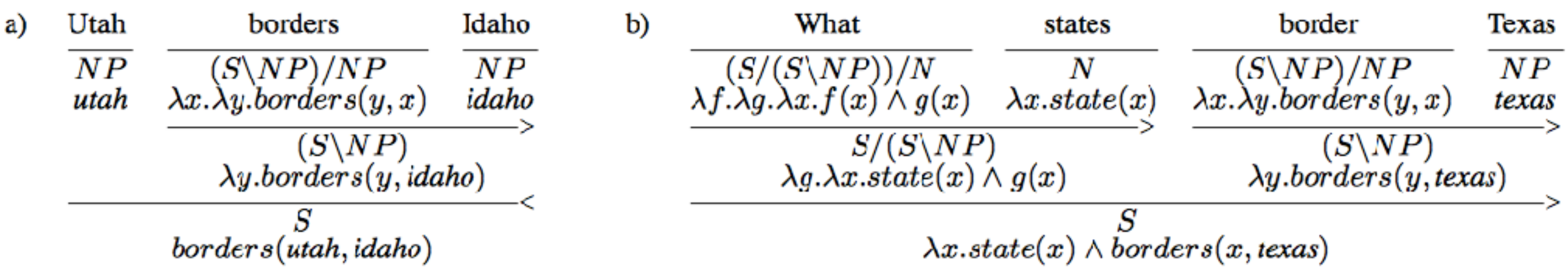
Syntax-driven Semantic Parsing

Syntax-driven Semantic Parsing

- Parse into syntax, then convert into meaning: no need to annotate meaning representation itself
- CFG → first order logic (e.g. Jurafsky and Martin 18.2)
- Dependency → first order logic (e.g. Reddy et al. 2017)
- Combinatory categorial grammar (CCG) → first order logic (e.g. Zettlemoyer and Collins 2012)

CCG and CCG Parsing

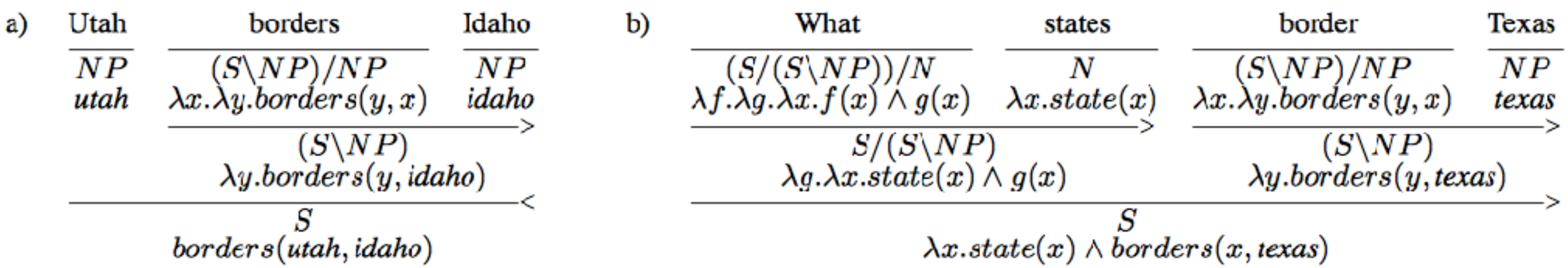
- CCG a simple syntactic formalism with strong connections to logical form
- Syntactic tags are combinations of elementary expressions (S, N, NP, etc)



- Strong syntactic constraints on which tags can be combined
- Much weaker constraints than CFG on what tags can be assigned to a particular word

Supertagging

- Basically, tagging with a very big tag set (e.g. CCG)

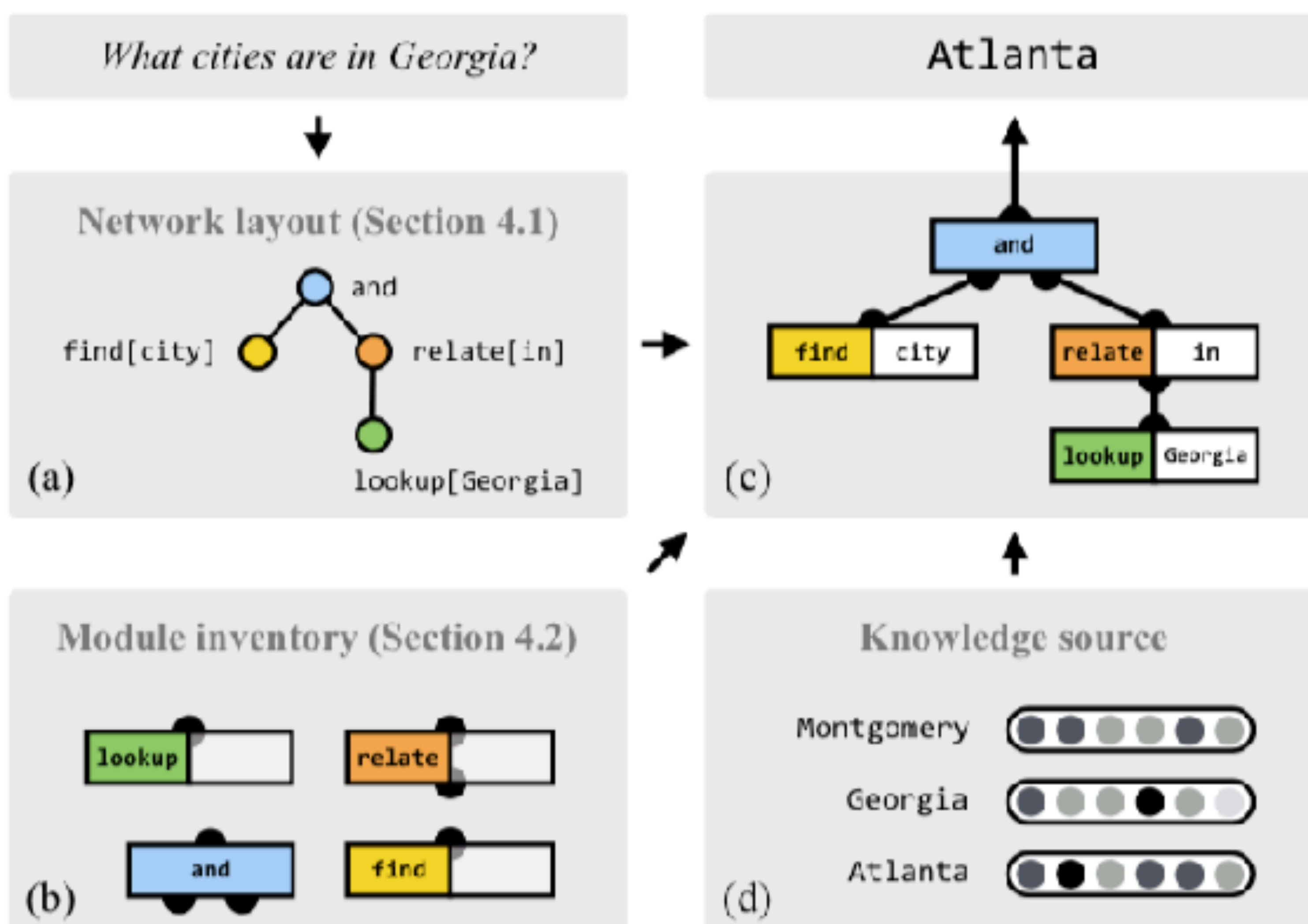


- If we have a strong super-tagger, we can greatly reduce CCG ambiguity to the point it is deterministic
- Standard LSTM taggers w/ a few tricks perform quite well, and improve parsing (Vaswani et al. 2017)
  - Modeling the compositionality of tags
  - Scheduled sampling to prevent error propagation

Neural Module Networks: Soft Syntax-driven Semantics

(Andreas et al. 2016)

- Standard syntax->semantic interfaces use symbolic representations
- It is also possible to use syntax to guide structure of neural networks to learn semantics



Shallow Semantics

Semantic Role Labeling
(Gildea and Jurafsky 2002)

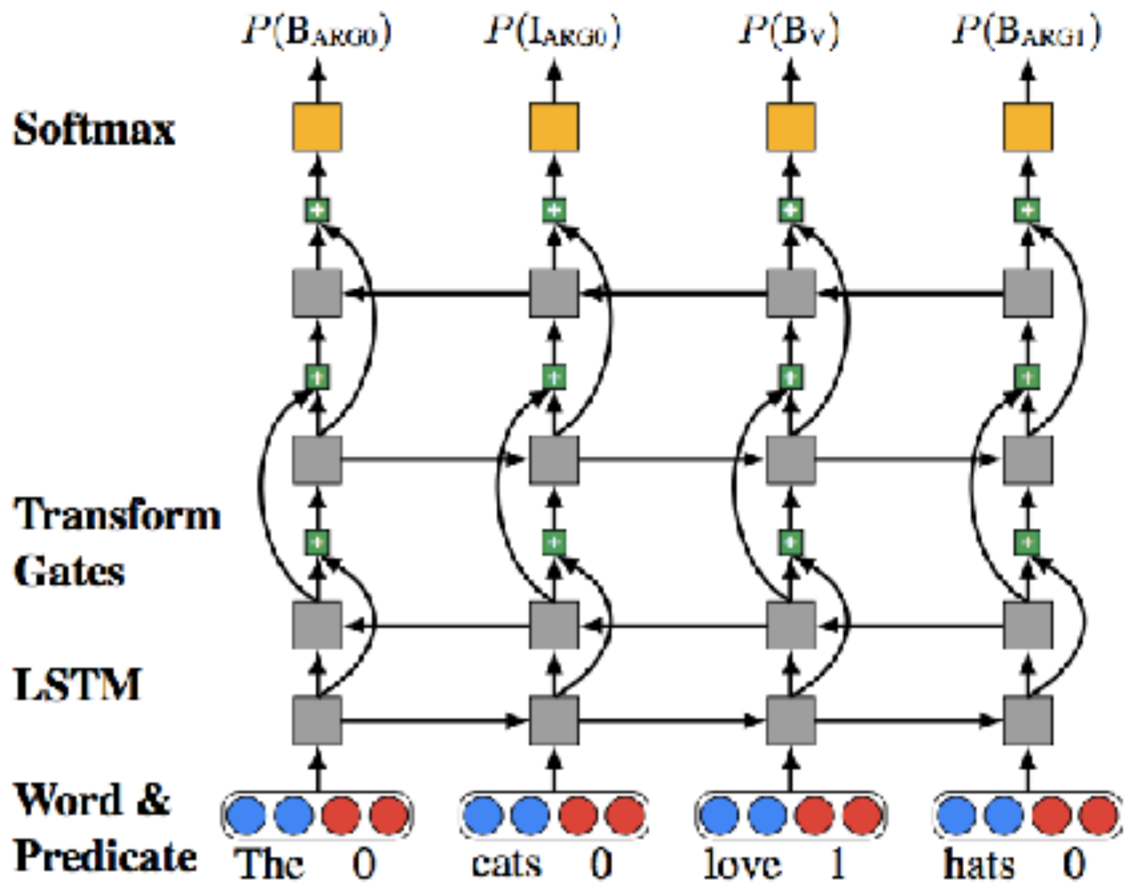
- Label “who did what to whom” on a span-level basis

(1) [$_{Judge}$ She] **blames** [$_{Evaluee}$ the Government] [$_{Reason}$ for failing to do enough to help] .

(2) [$_{Message}$ “I’ll knock on your door at quarter to six”] [$_{Speaker}$ Susan] **said**.

Neural Models for Semantic Role Labeling

- Simple model w/ deep highway LSTM tagger works well (Le et al. 2017)



- Error analysis showing the remaining challenges

Questions?