Transition-based AMR Graph Parser

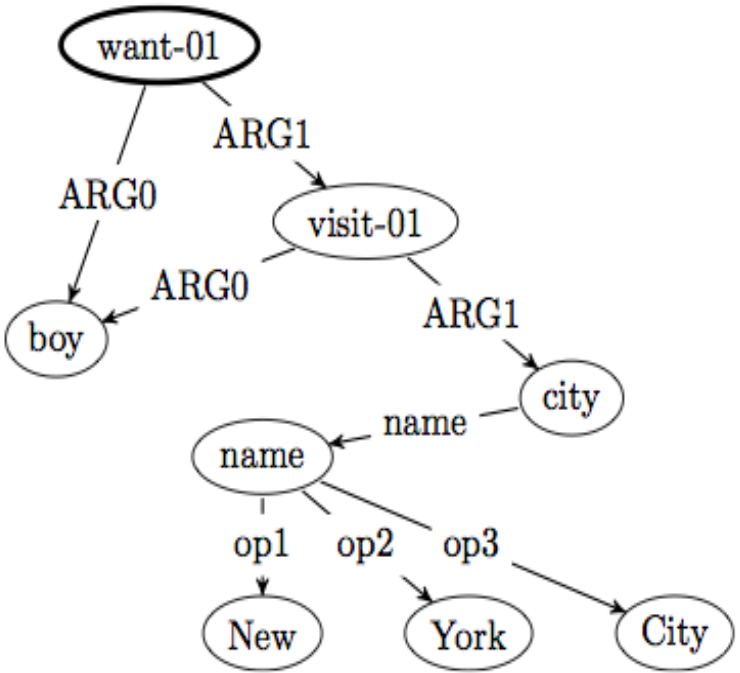
July.08.2014

Transition-based AMR Graph Parser

- AMR Graph
- Preprocessing
  - Alignment
  - Gold graph generation
- Dependency tree to Graph transition system
- Problems

AMR Graph

```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (g / visit-01
          :ARG0 b
          :ARG1 (c / city
                  :name (n / name
                          :op1 New
                          :op2 York
                          :op3 City))))
```



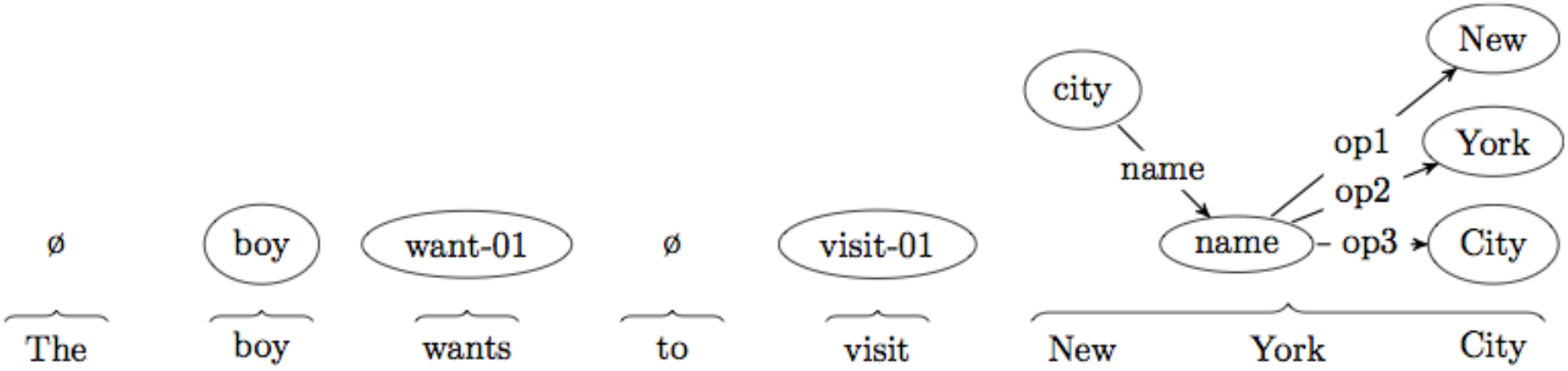
Alignment

Why alignment?

```
# ::id lpp_1943.5 ::date 2012-06-07T17:06:32 ::annotator ISI-AMR-05 ::
preferred
# ::snt In the book it said : " Boa constrictors swallow their prey
whole , without chewing it .
# ::zh 这本书中写道:“这些蟒蛇把它们的猎获物不加咀嚼地囫囵吞下
# ::save-date Fri Sep 6, 2013 ::file lpp_1943_5.txt
(s2 / say-01
      :ARG0 (b2 / book)
      :ARG1 (s / swallow-01
            :ARG0 (b / boa
                  :mod (c / constrictor))
            :ARG1 (p / prey
                  :mod (w / whole)
                  :poss b)
            :manner (c2 / chew-01 :polarity -
                  :ARG0 b
                  :ARG1 p)))
```

Alignment

Rule-base Automatic Alignment (Flanigan,2014)



For each concept in the amr graph, it greedily searches the sentence for corresponding span of words using a list of rules.

Gold graph generation

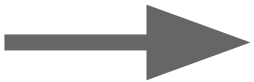
Alignment + AMR graph = Gold graph

1. each node in gold graph represents one of the spans in the sentence

2. each node also contains its entity tag in the AMR graph.

e.g.

```
(c / city
  :name (n / name
           :op1 New
           :op2 York
           :op3 City))))
```



```
node {
     start: 6
     end: 8
     ENTITY_TAG:CITYNE
 }
```

Dependency Tree to Graph Transition System

A transition system is a quadruples S = (C, T, C_s,C_t), where

1. C is a set of parsing states (configurations)
2. T is a set of parsing actions (transitions), each of which is a function t: C_i->C_j
3. C_s is an initialization function, mapping a sentence and its dependency tree to an initial parsing state
4. C_t is a set of terminal parsing states

Parsing State

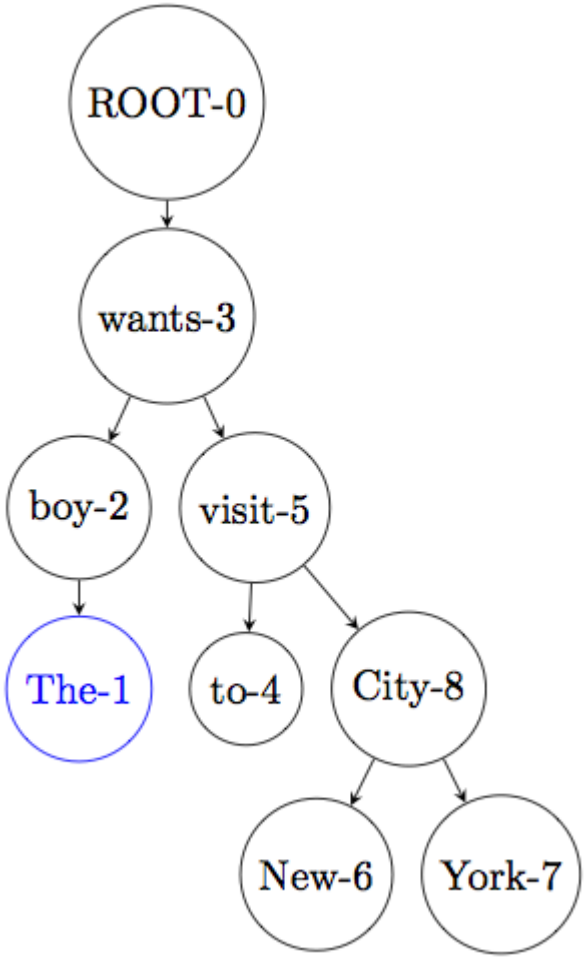
- each parsing state is a triple c = (σ,β,A), where
  a. σ is a buffer of nodes initialized by bottom-up traversal of current sentence's dependency tree dT, with buffer top i
  b. β is a buffer of nodes which are children of current σ top i, with buffer top j
  c. A is a partially parsed graph initialized with dependency tree dT
- terminal state is ([],[],A)

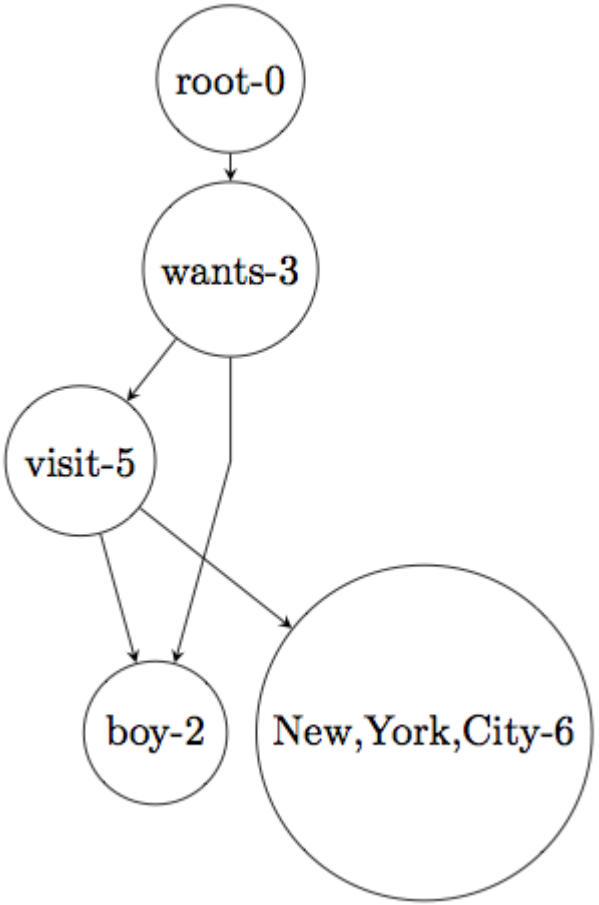
Parsing Action

- ➔ if β is not empty:
  - delete edge: (i|σ,j|β,A) => (i|σ,β,A.remove_edge(i,j))
  - swap: (i|σ,j|β,A) => (i|j|σ,β,A.swap(i,j))
  - replace head: (i|σ,j|β,A) => (j|σ,β=[i's children except j],A.replace_head(i,j))
  - merge: (i|σ,j|β,A) => ((i<j)?i:j|σ,(i<j)?j:i|β,A.merge(i,j))
  - next: (i|σ,j|β,A) => (i|σ,β,A) # correct edge
- ➔ β is empty:
  - add child k: (i|σ,[],A) => (i|σ,[],A.add_edge(i,k))
  - finish: (i|σ,[],A) => (t|σ,β_t,A) # done with current node
  - delete node: (i|σ,[],A) => (t|σ,β_t,A.delete_node(i))

Example

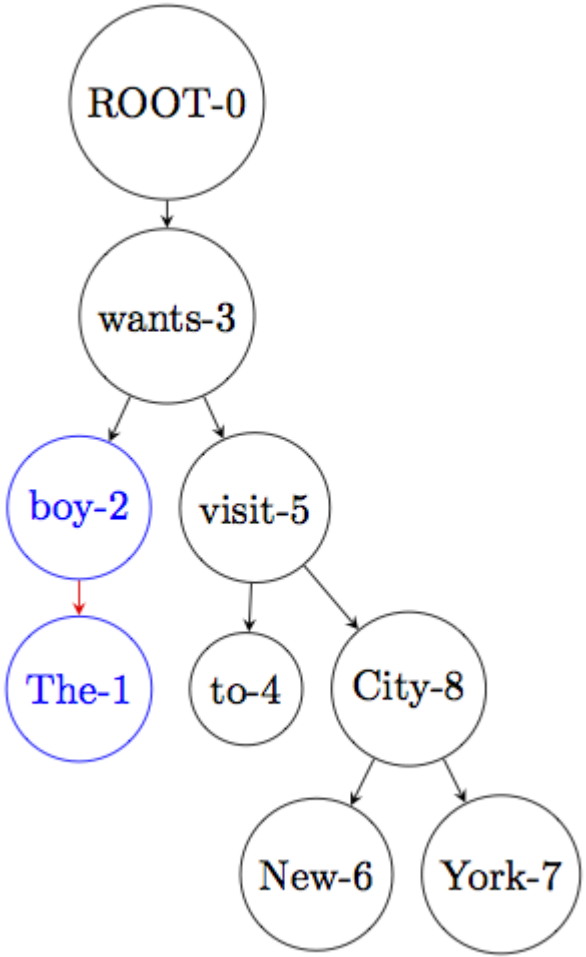
σ:[The-1|σ]
β:[]
Finish

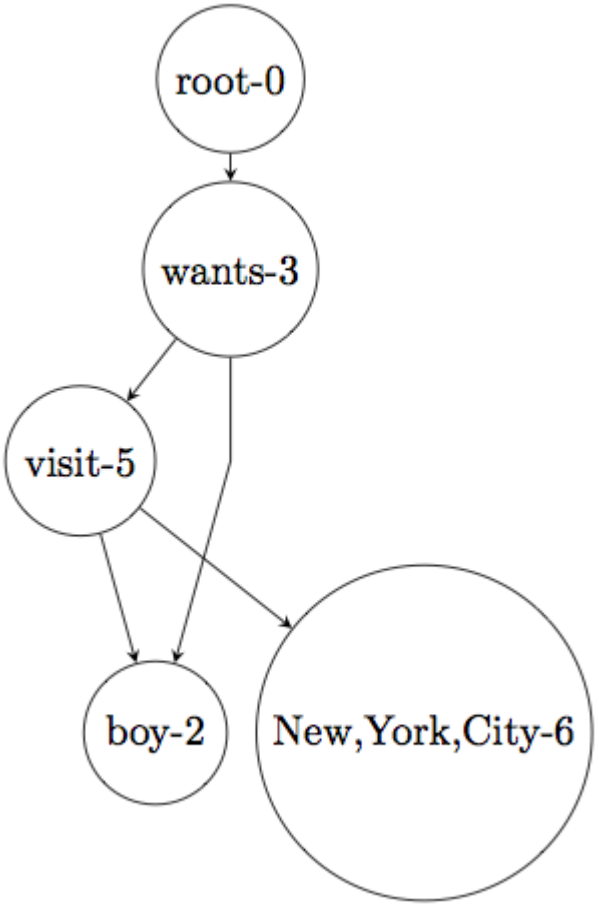




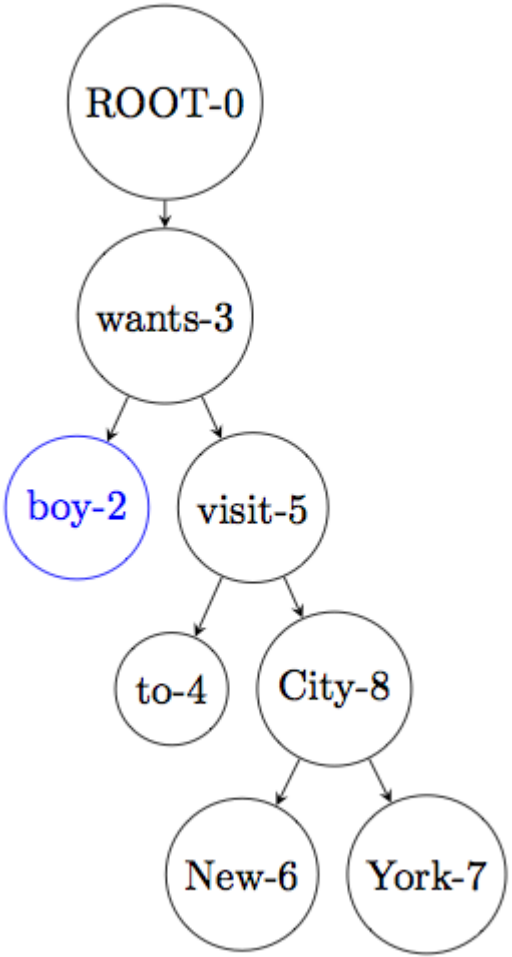
Example

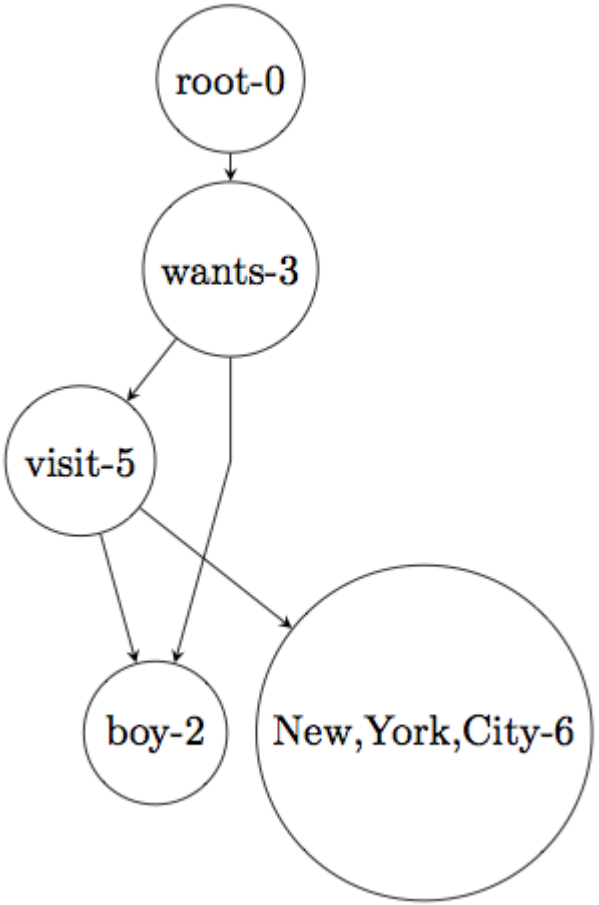
σ:[boy-2|σ]
β:[The-1]
Delete child





Example

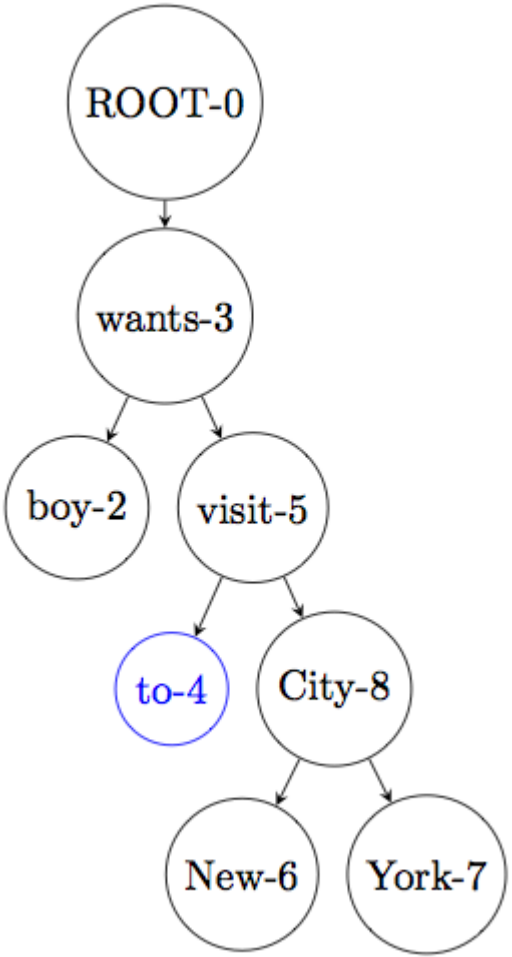


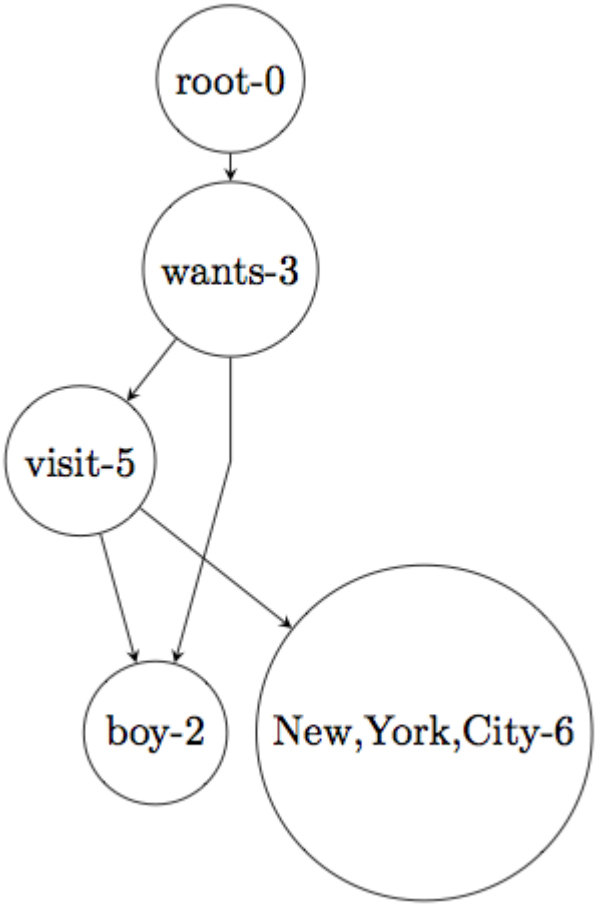


σ:[boy-2|σ]
β:[]
Finish

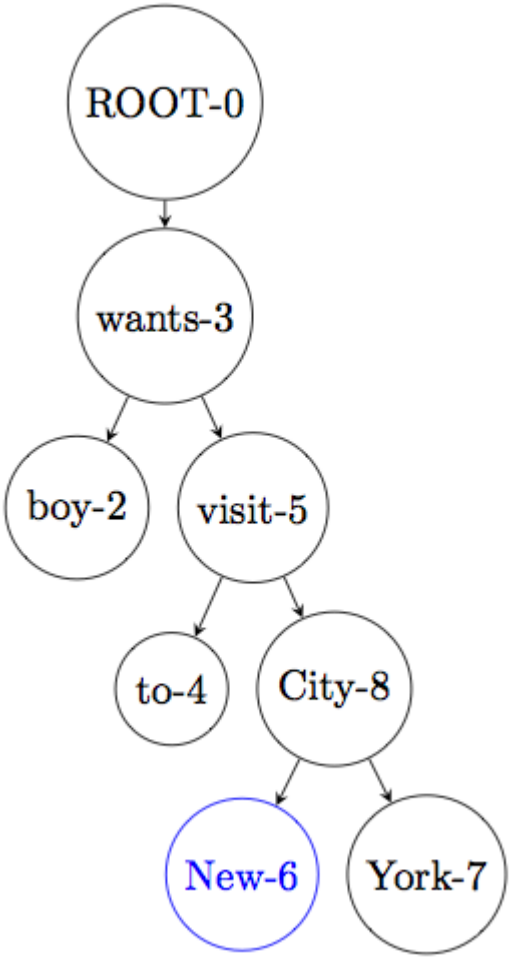
Example

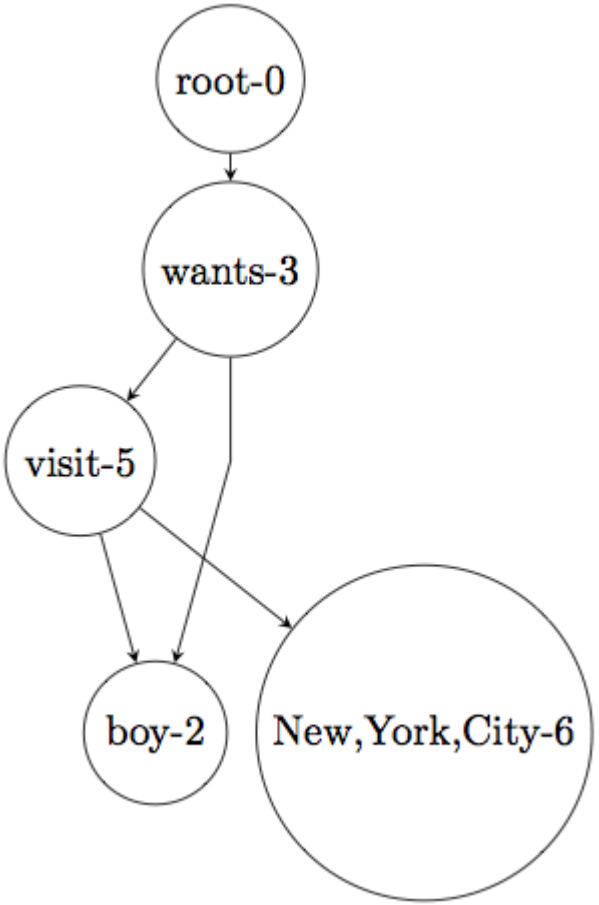
σ:[to-4|σ]
β:[]
Finish





Example

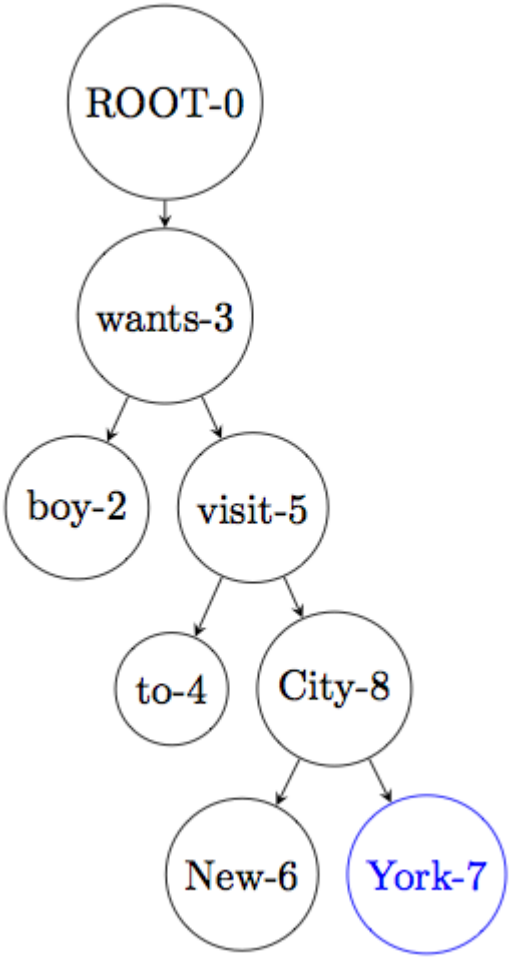


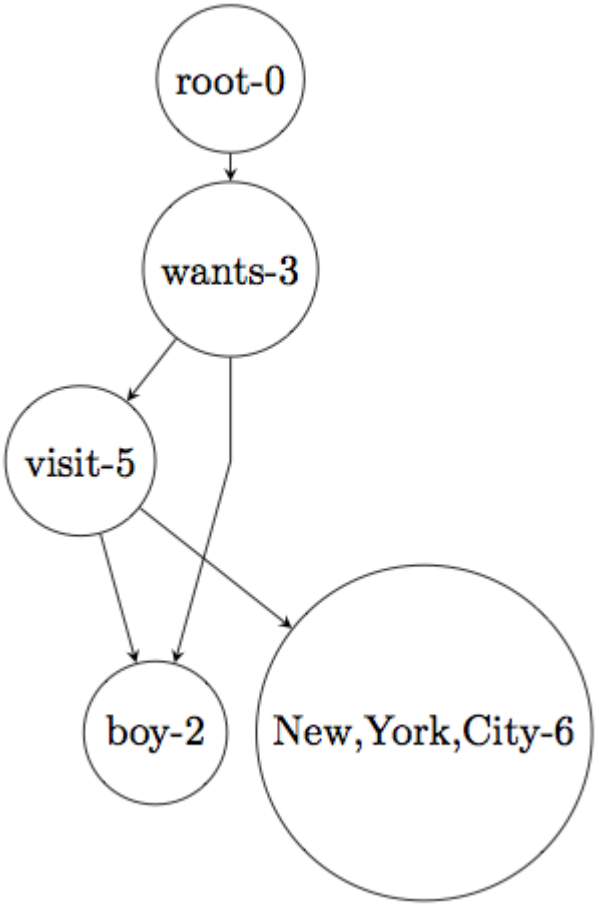


σ:[New-6|σ]
β:[]
Finish

Example

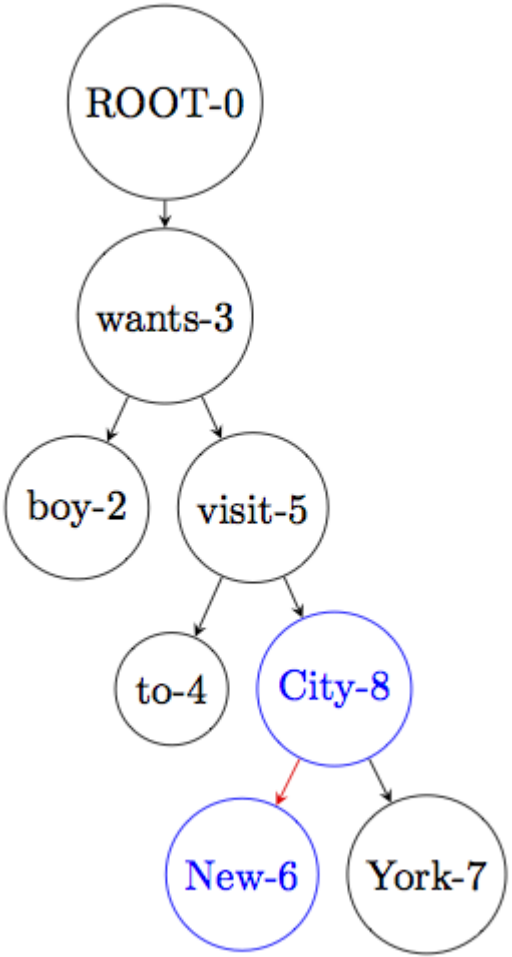
σ:[York-7|σ]
β:[]
Finish

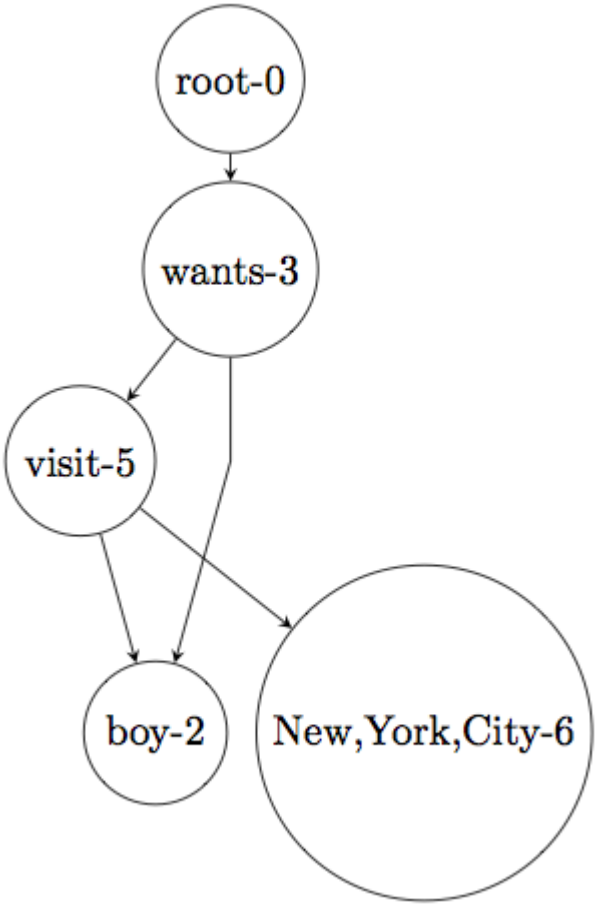




Example

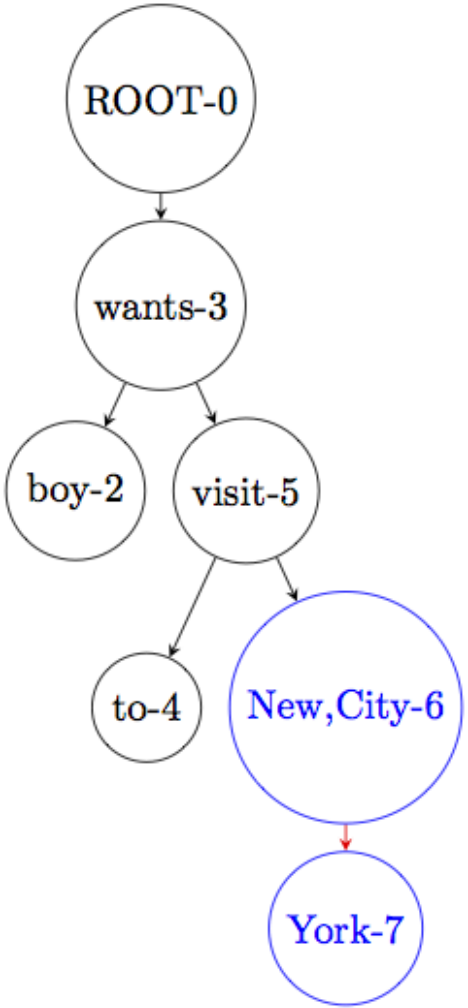
σ:[City-8|σ]
β:[New-6|β]
Merge

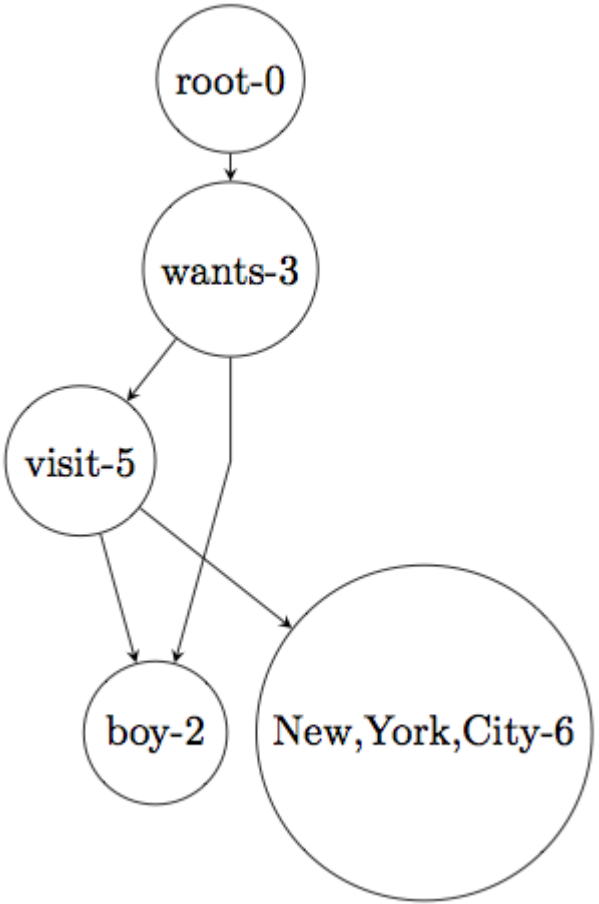




Example

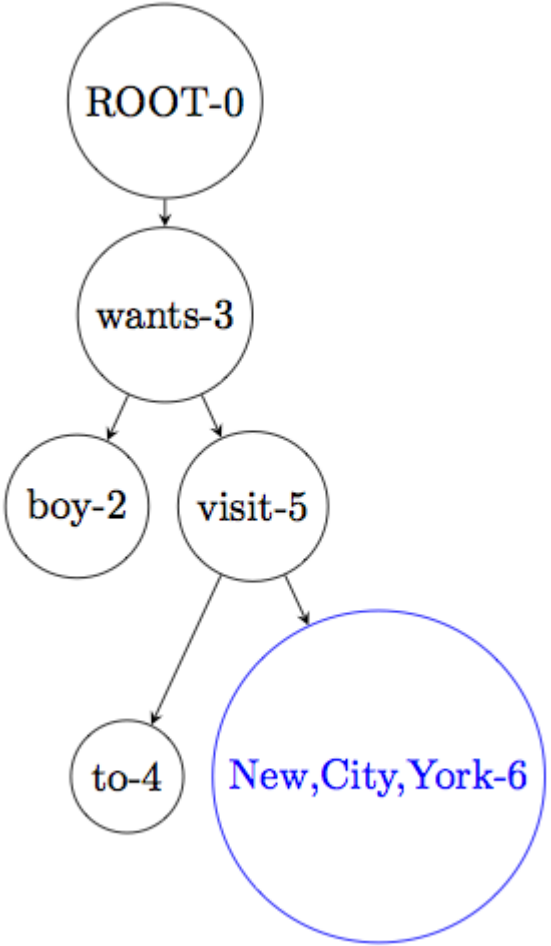
σ:[New,City-6|σ]
β:[York-7]
Merge

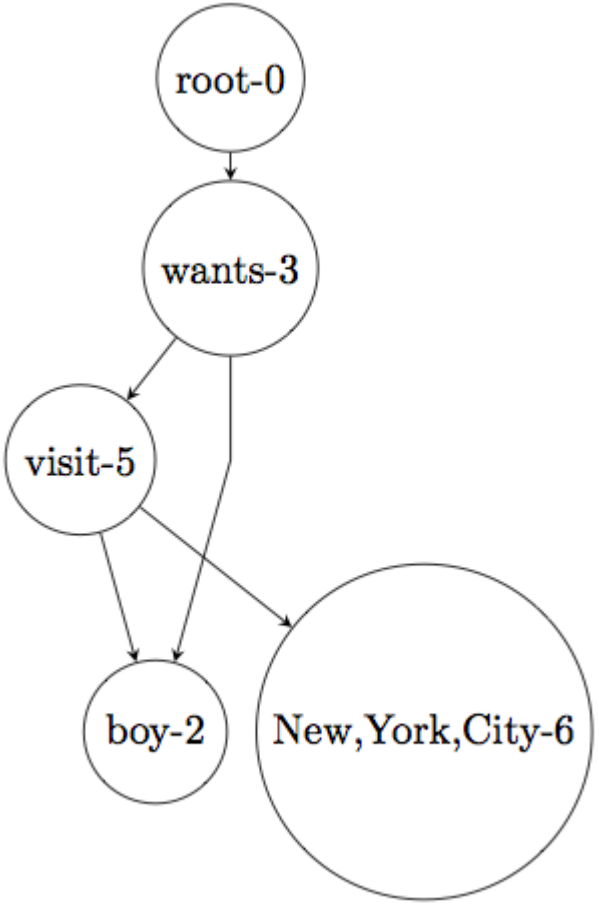




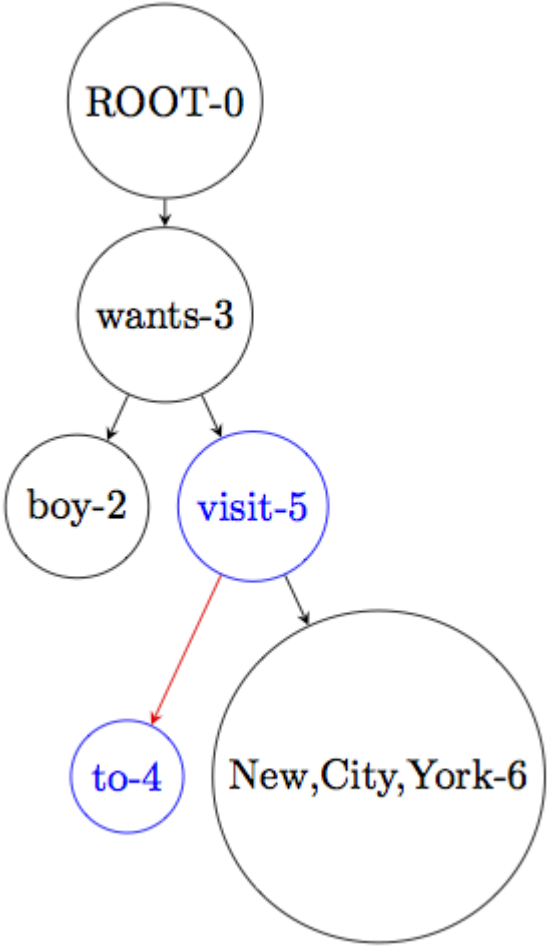
Example

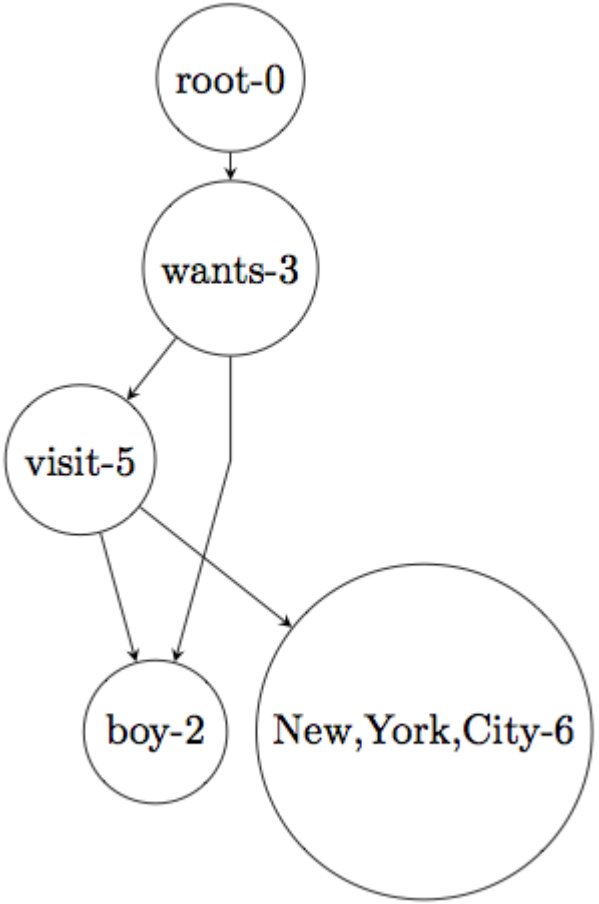
σ:[New,York,
City-6|σ]
β:[]
Finish





Example

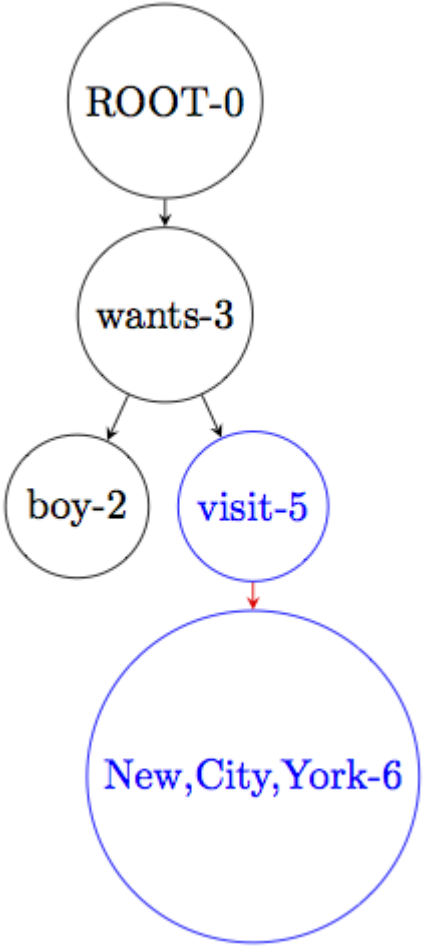


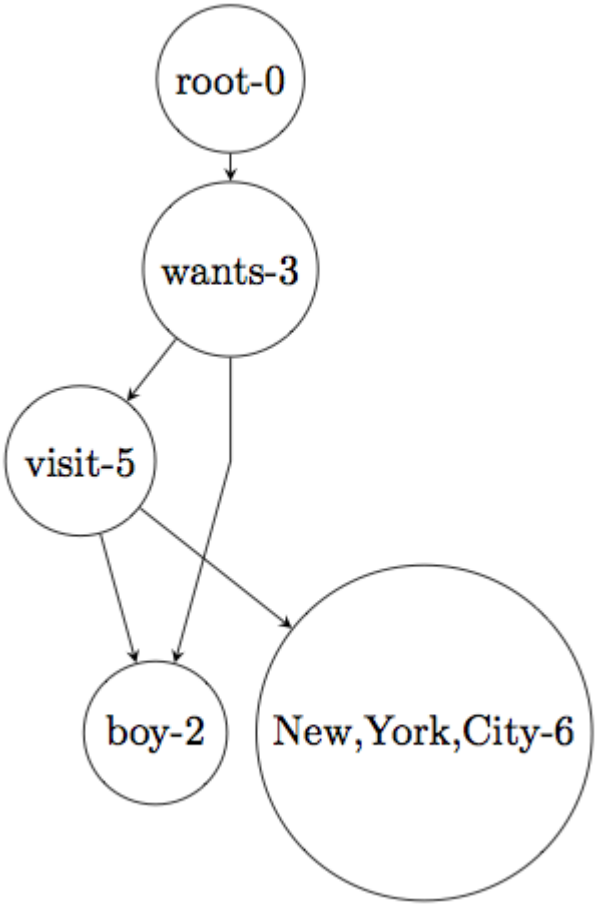


σ:[visit-5|σ]
β:[to-4|β]
Delete child

Example

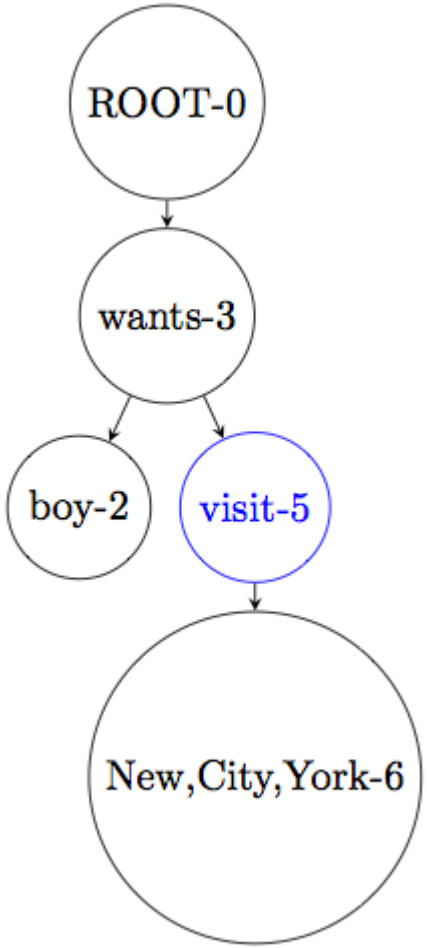
σ:[visit-5|σ]
β:[New,York,
City-6]
Next

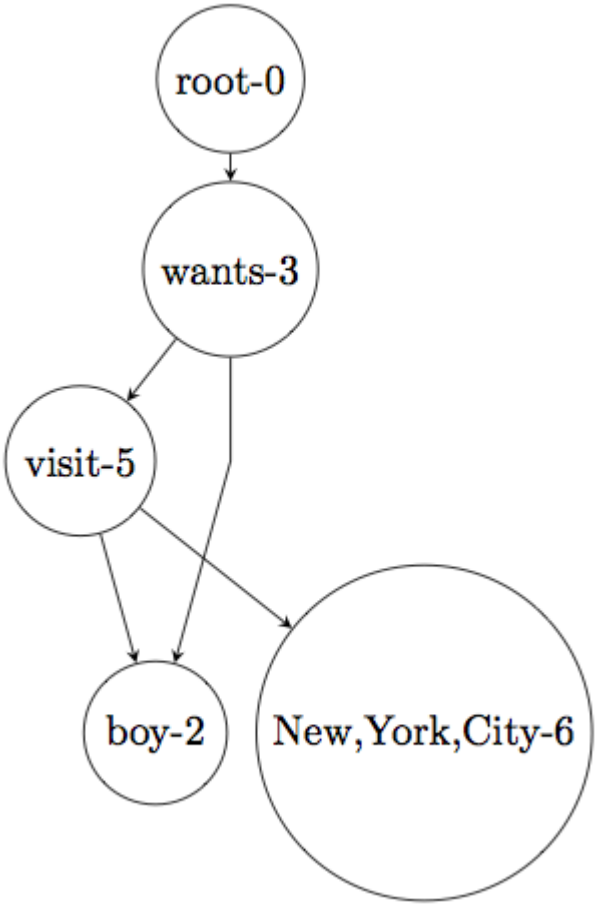




Example

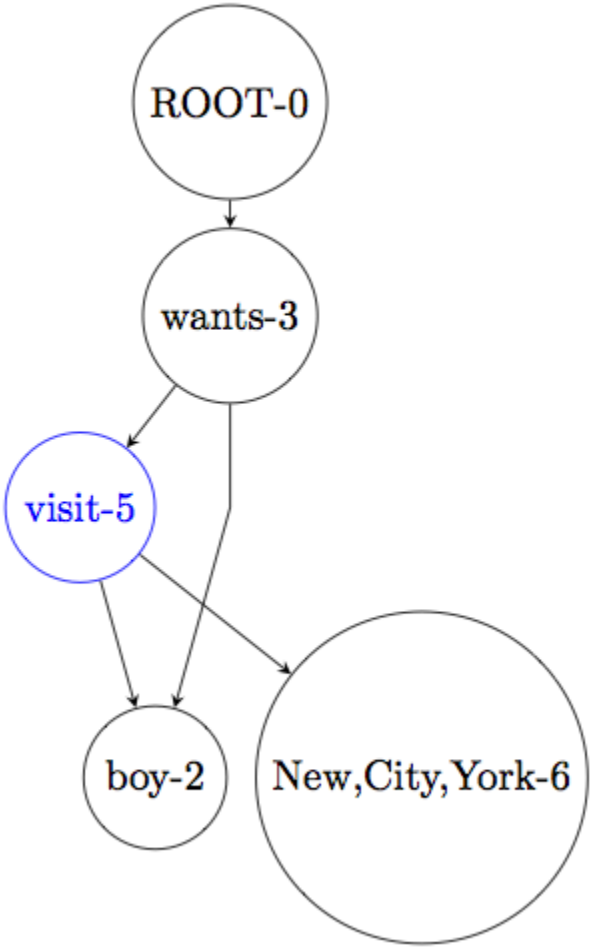
σ:[visit-5|σ]
β:[]
Add child
boy-2

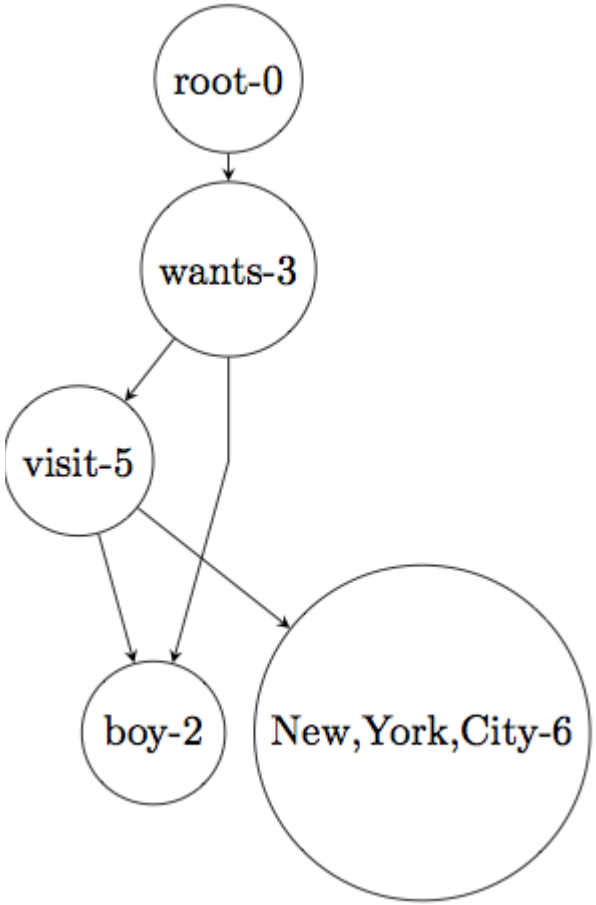




Example

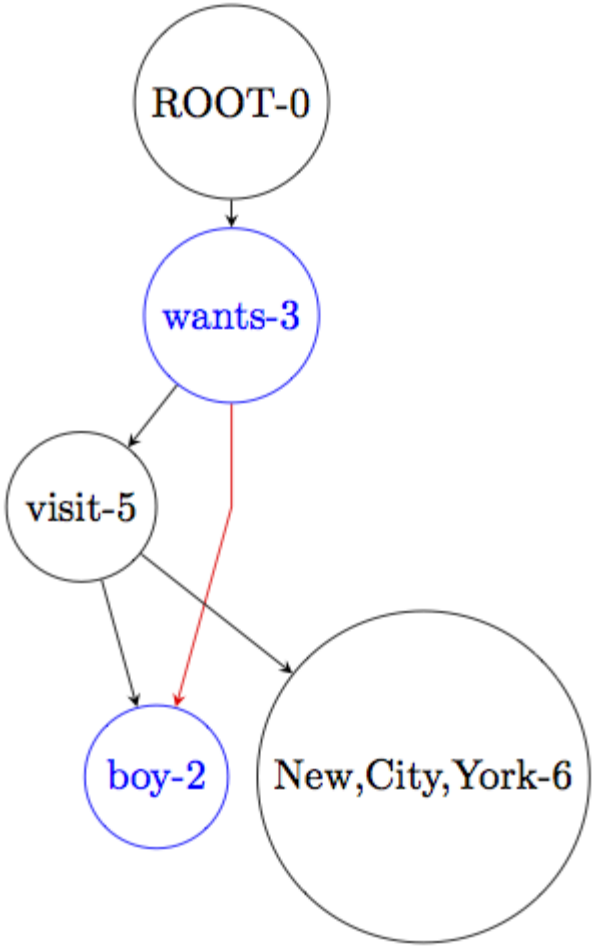
σ:[visit-5|σ]
β:[]
Finish

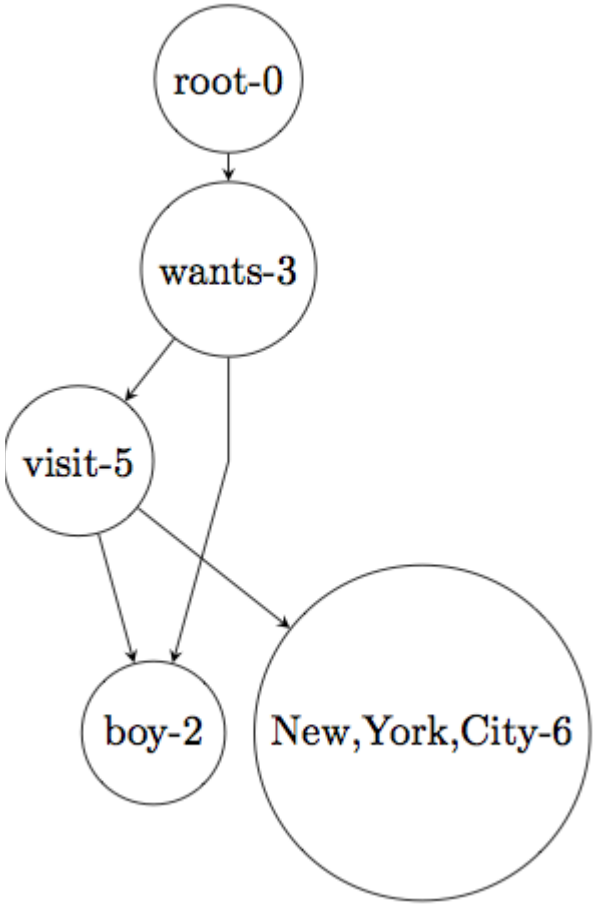




Example

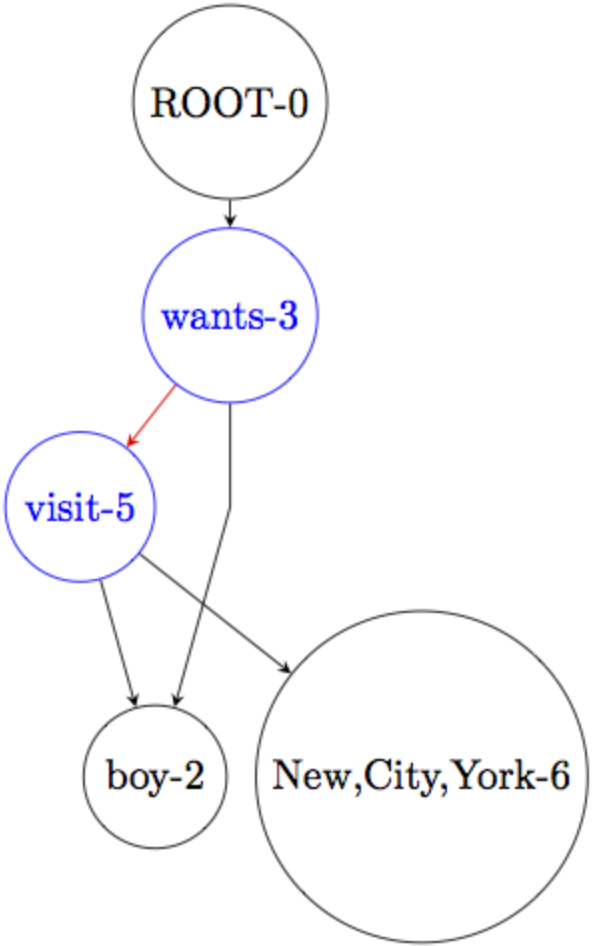
σ:[wants-3|σ]
β:[boy-2|β]
Next

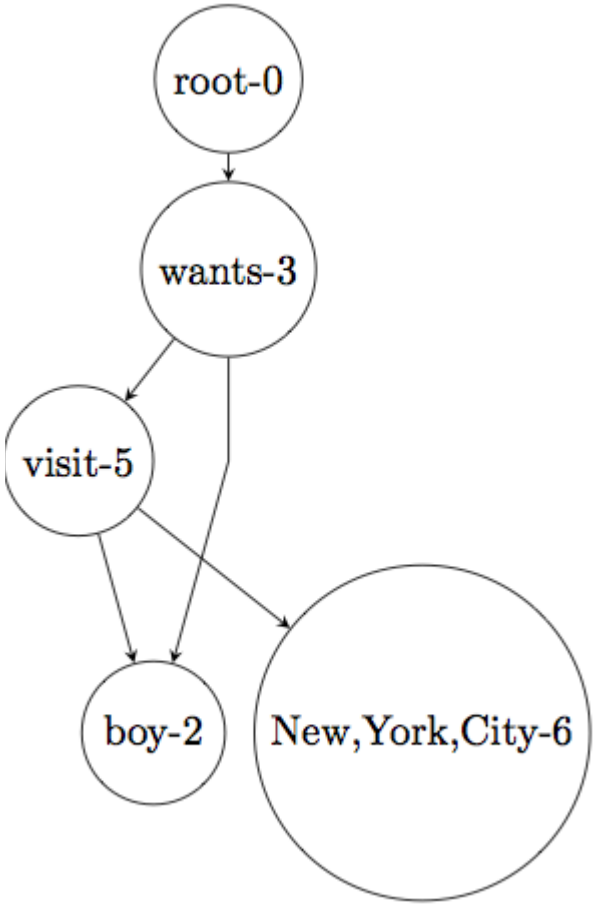




Example

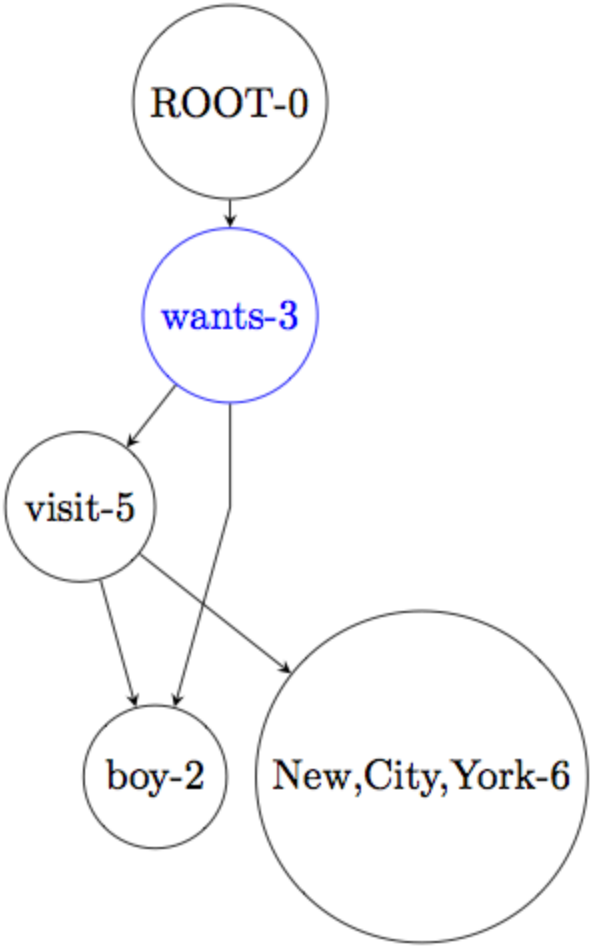
σ:[wants-3|σ]
β:[visit-5]
Next

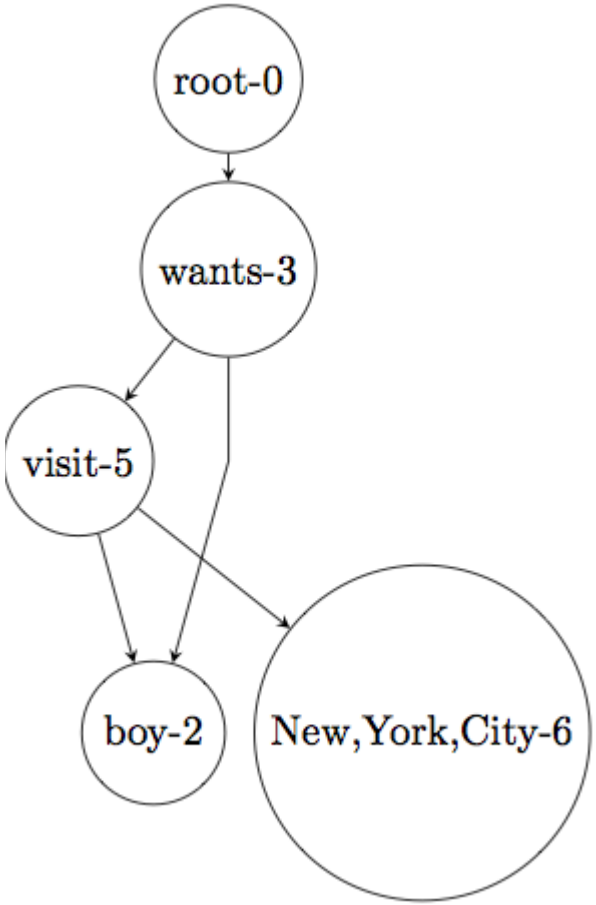




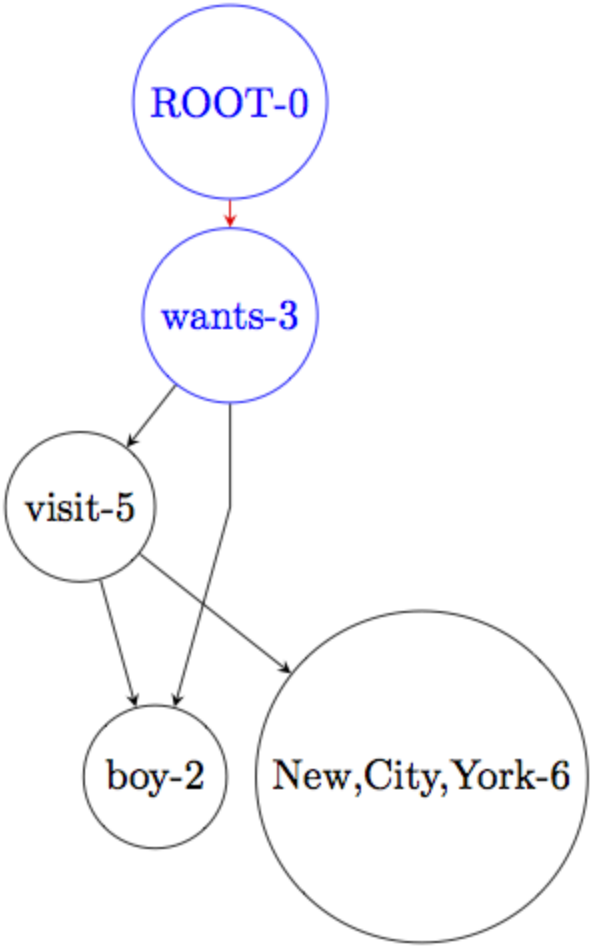
Example

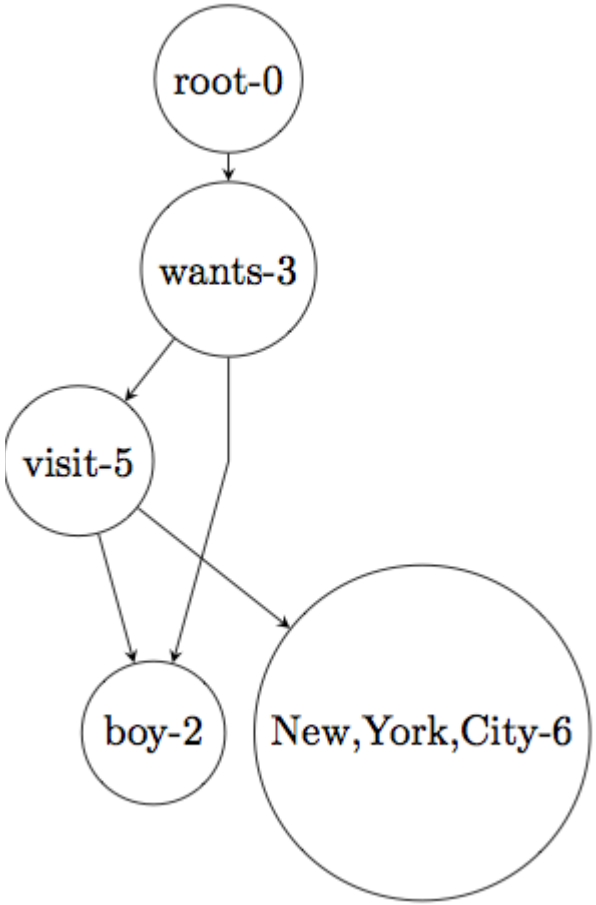
σ:[wants-3|σ]
β:[]
Finish





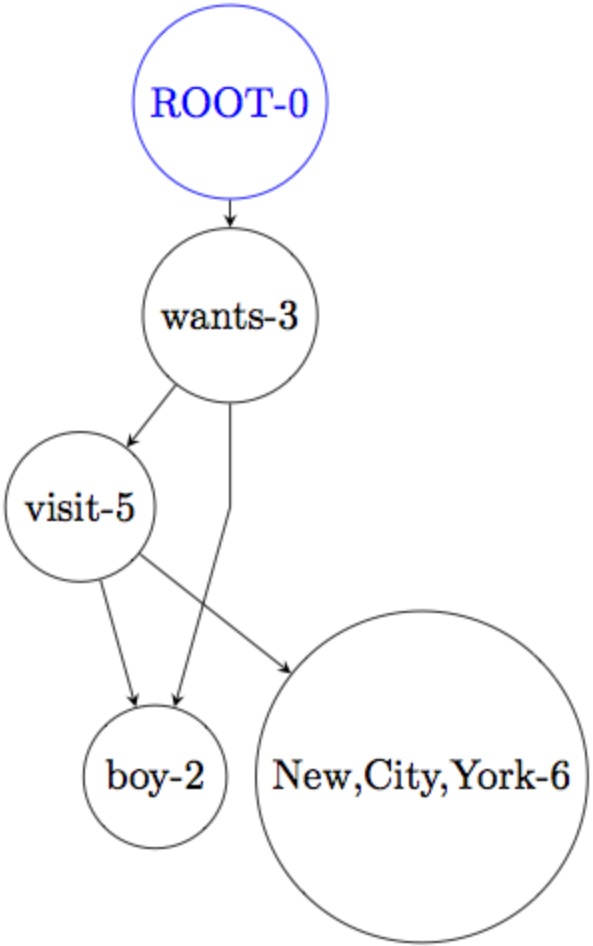
Example

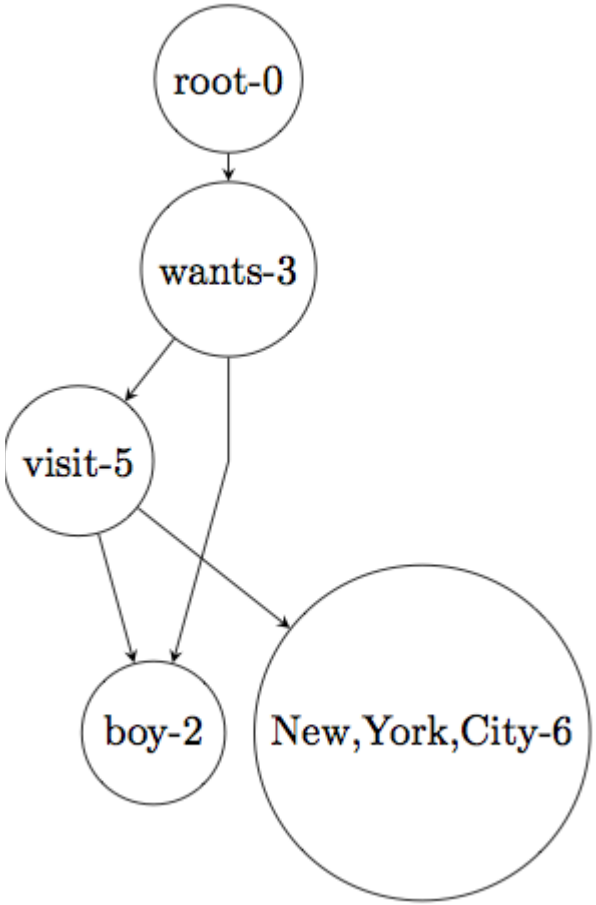




σ:[Root-0|σ]
β:[wants-3|β]
Next

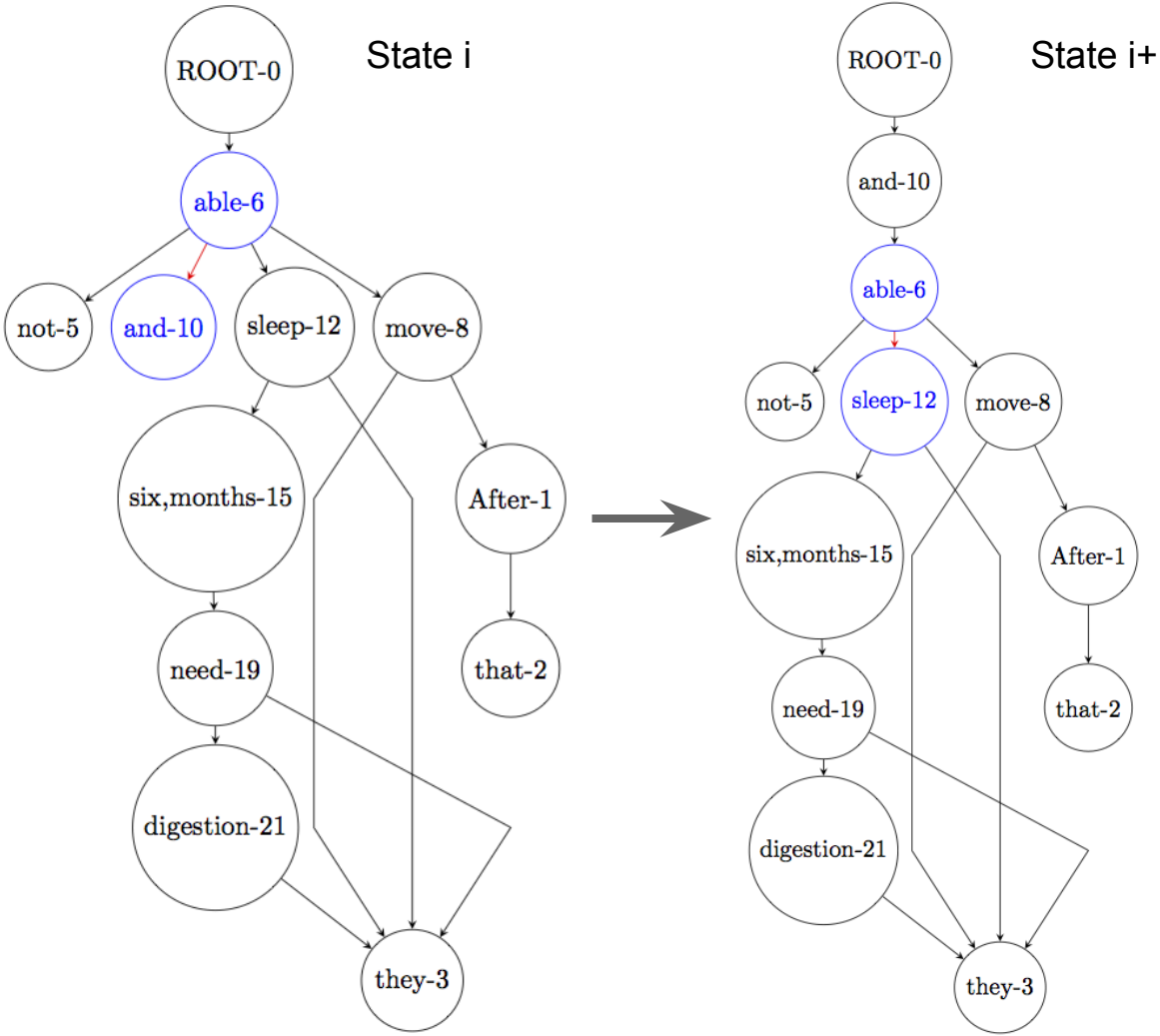
Example


ROOT-0
wants-3
visit-5
boy-2
New,City,York-6


root-0
wants-3
visit-5
boy-2
New,York,City-6

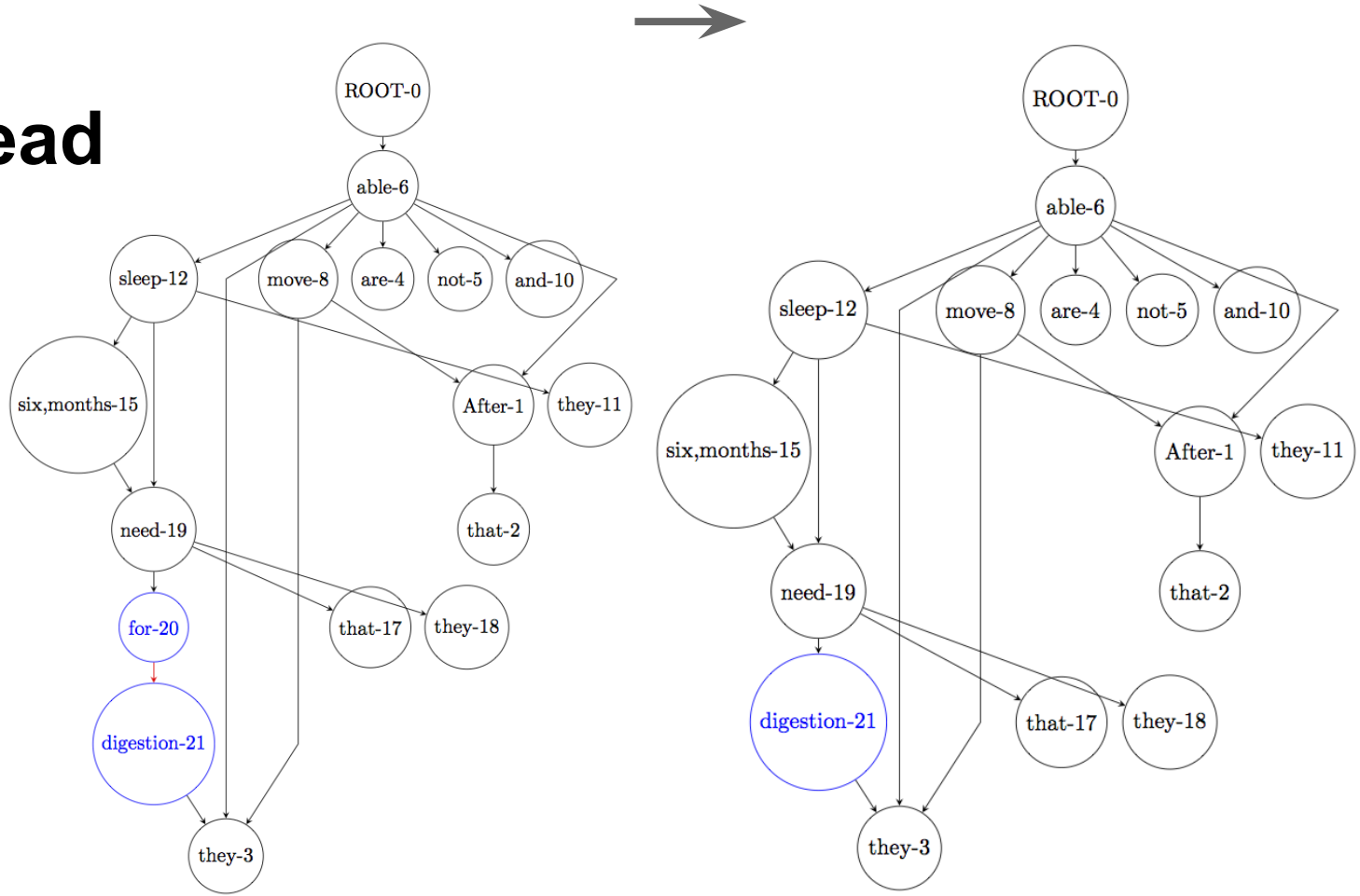
Example
Swap

σ:[able-6|σ]
β:[and-10|β]
Swap



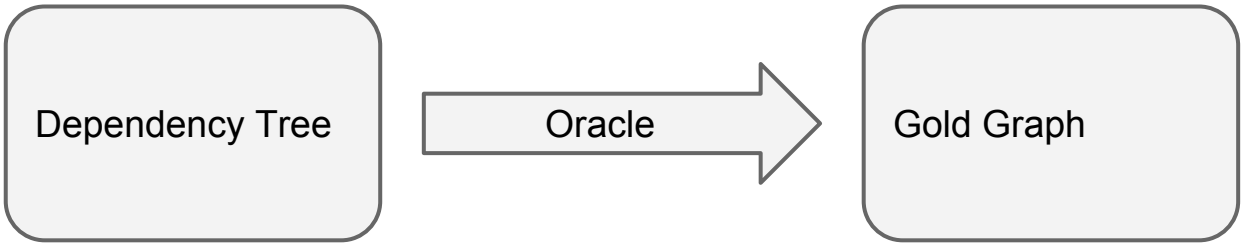
Example
Replace head

σ:[for-20|σ]
β:[digestion-21|β]
Replace head

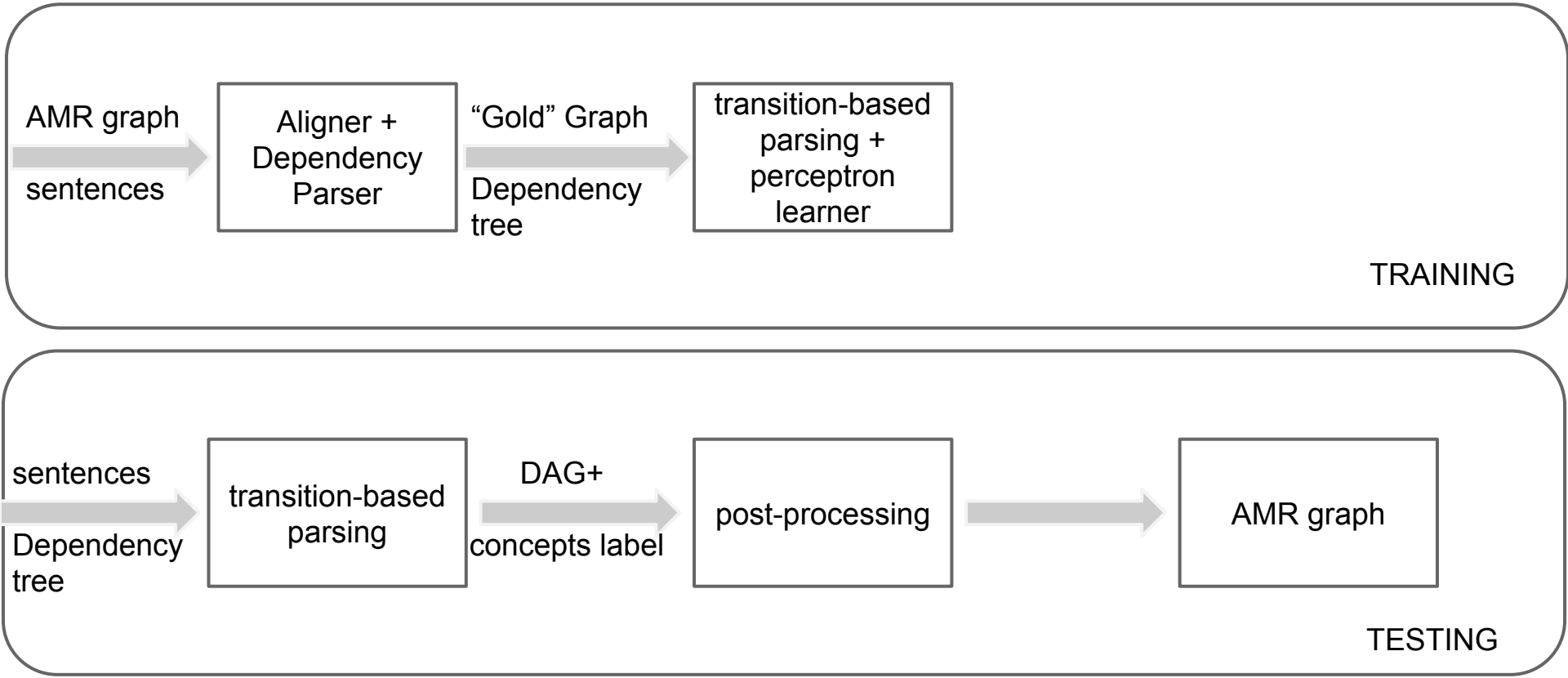


Deterministic Parsing

If we assume alignment is correct, which means we always get the gold graph (alignment + AMR graph), we can design an oracle to generate unique sequence of parsing actions for each sentence in corpus.



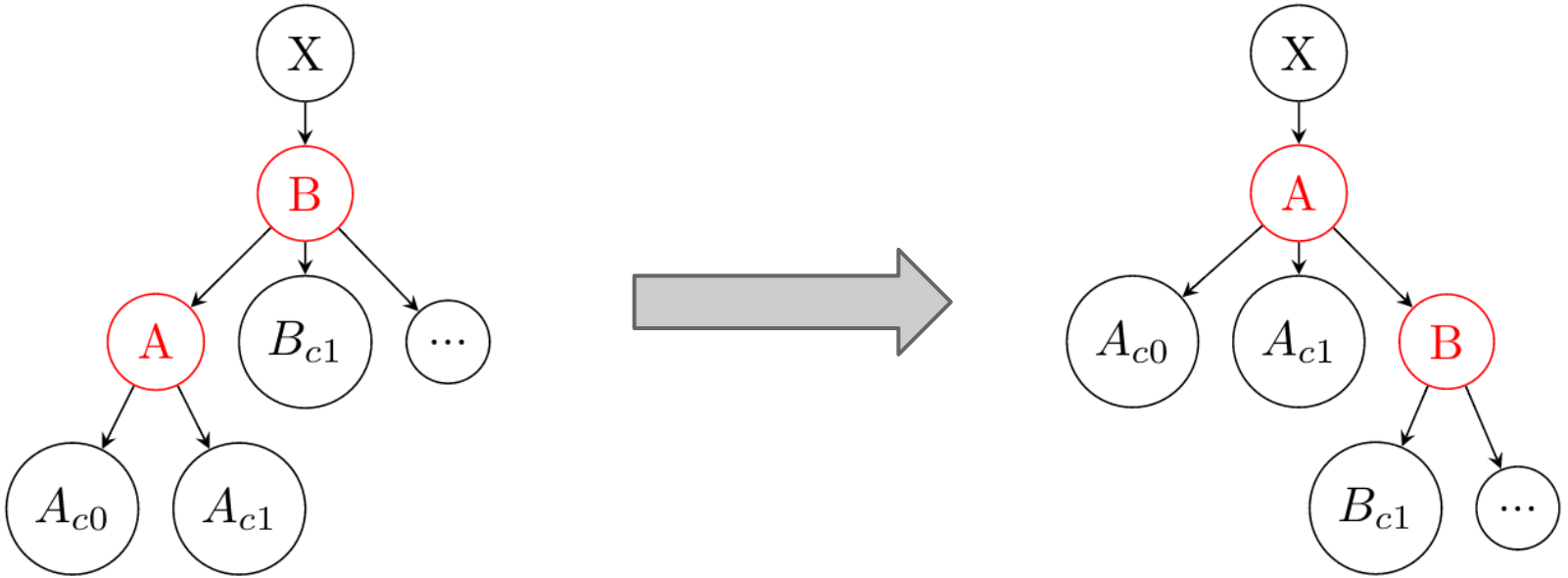
Pipeline


AMR graph
sentences
Aligner + Dependency Parser
“Gold” Graph
Dependency tree
transition-based parsing + perceptron learner
TRAINING
sentences
Dependency tree
transition-based parsing
DAG+
concepts label
post-processing
AMR graph
TESTING

Problems

1. Some of the empirically designed actions should be subcategorized

SWAP: currently only reverse the relation B-> A and all B's parents should point to A



SWAP- Example

DOMINANT: all the current node's children should be reattached to new head

It was a picture of a boa constrictor in the act of swallowing an animal .

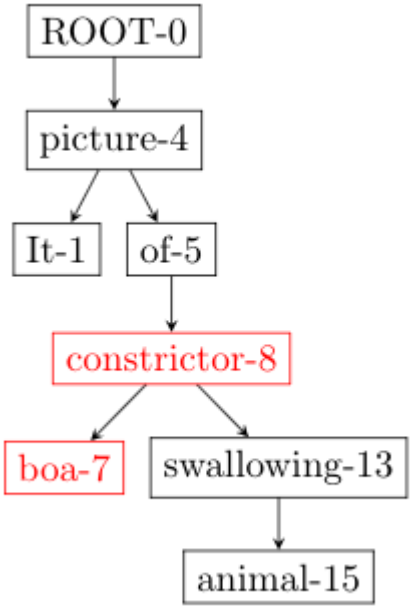


Fig1. partial parsed graph

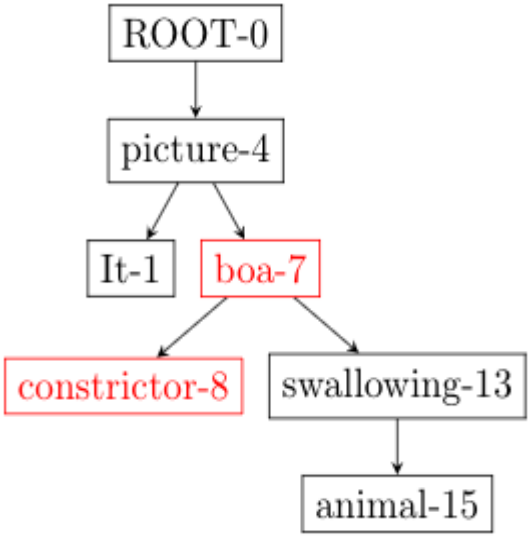


Fig2. gold graph

SWAP- Example

DOMINANT: all the current node's children should be reattached to new head

I would talk to him about bridge , and golf , and politics , and neckties .

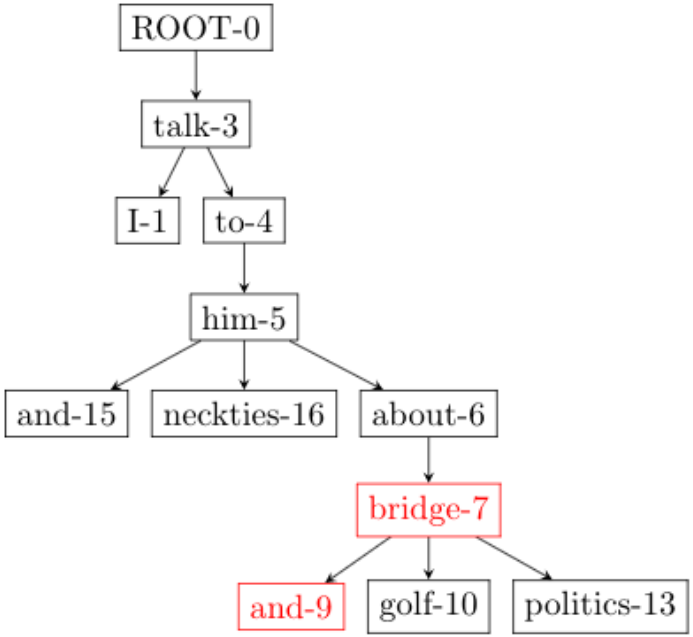


Fig1. partial parsed graph

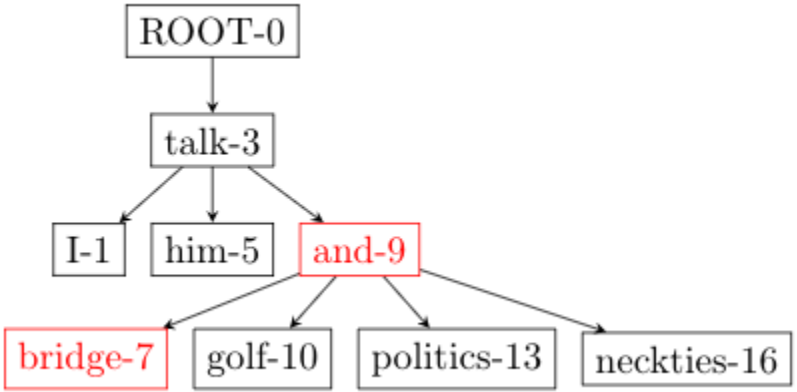


Fig2. gold graph

SWAP- Example

PARTIAL REATTACH

After that they are not able to move , and they sleep through the six months that they need for digestion . "

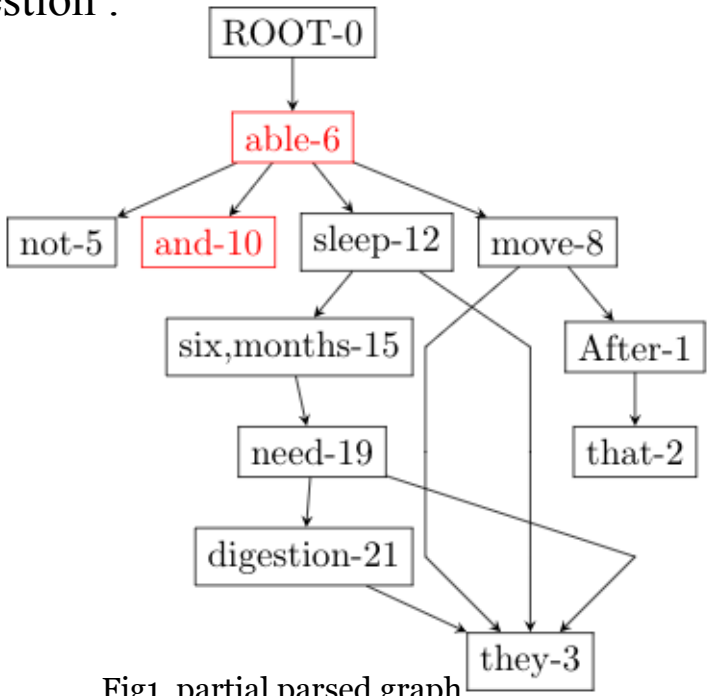


Fig1. partial parsed graph

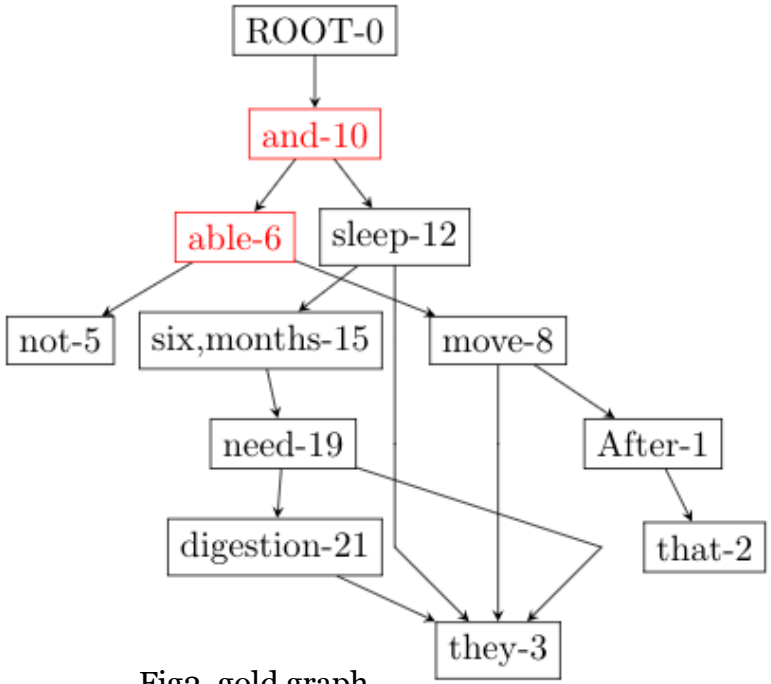


Fig2. gold graph

SWAP- Example

TRIVIAL: just reverse the direction of current relation, no current node's children follow new head

At a glance I can distinguish China from Arizona .

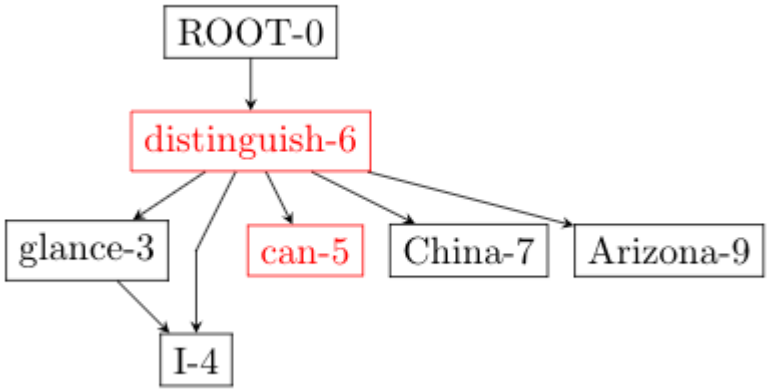


Fig1. partial parsed graph

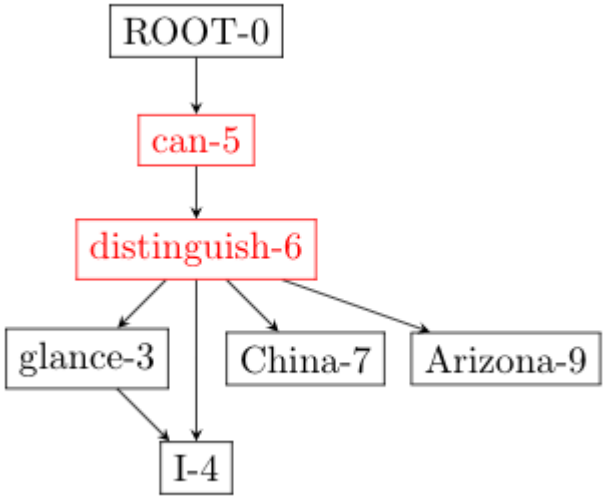


Fig2. gold graph

Problems

2. Not like shift-reduce parser, in which oracle is given with no extra effort; it is difficult to design a consistent oracle in this scenario.

Sometimes we would like to delay some of the action and do others first.

Example

Current oracle will do action DELETE-EDGE for (disobey|σ,overpowering|β) first;
Intuitively, it would be nice if we delay the DELETE-EDGE decision and do SWAP (trivial) for (disobey|σ,dare|β) first.

When a mystery is too overpowering , one dare not disobey .

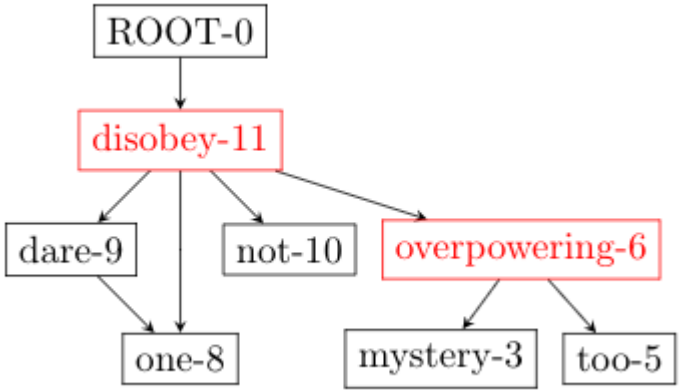


Fig1. partial parsed graph

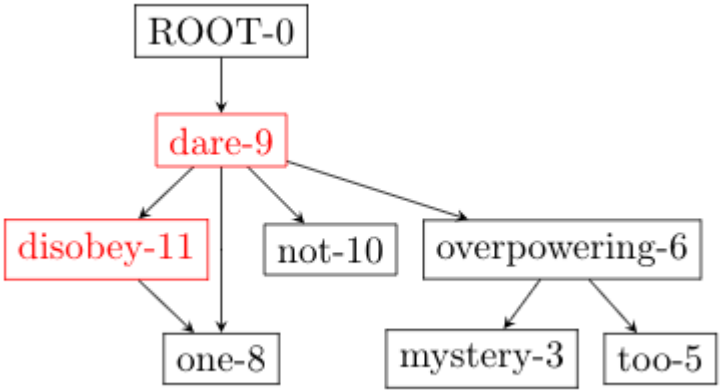


Fig2. gold graph

Experiment result

Currently we don’t have edge label and concept label, we can measure the performance by looking only the attachment.

$$\text{Precision} = \frac{num\_of\_correct\_attachment\_both\_in\_parsed\_graph\_gold\_graph}{num\_of\_attachment\_in\_parsed\_graph}$$

$$\text{Recall} = \frac{num\_of\_correct\_attachment\_both\_in\_parsed\_graph\_gold\_graph}{num\_of\_attachment\_in\_gold\_graph}$$

Experiment result

1. Similarity of attachment between dependency tree and gold graph?

`Precision:0.29 Recall:0.49 F1:0.37`

measure on 1250 sentence in AMR bank v1.2 “the little prince”

2. Using only word form, pos tag and name entity tag as features, We run 19 iteration on the 1250 training AMR sentences and the result on develop set (312 sentences) is:

`Precision:0.45 Recall:0.53 F1:0.48`

Problems

Alignment problem

Although the alignment has overall high accuracy (~90% F1) (Flanigan,2014), some errors can be severe and greatly harm the later gold graph generation.

1) concept not aligned
2) concept alignment error

graph generated from the corpus may be incorrect (disconnected or wrong connection). As a result, the training instances generated (sentence and its parsing action sequence) would have a lot of noise.

Problems

Alignment Problem

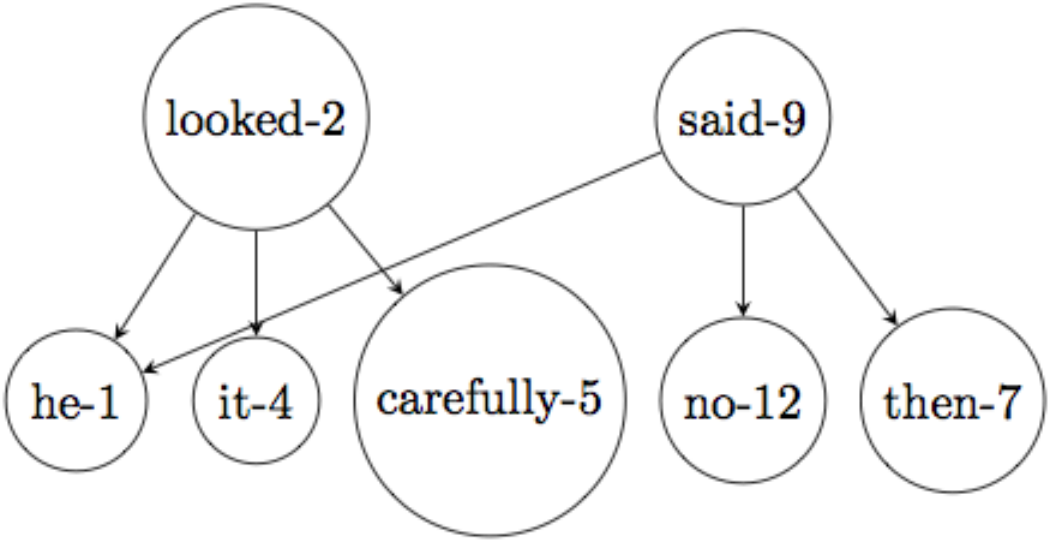
1). concept not aligned

```
# ::id lpp_1943.76 ::date 2012-07-24T13:26:32 ::annotator ISI-AMR-05 ::
preferred
# ::snt He looked at it carefully , then he said : " No .
# ::zh 他专心地看着, 随后又说:“我不要,
# ::save-date Sat Dec 29, 2012 ::file lpp_1943_76.txt
(a / and
  :op1 (l / look-01
         :ARG0 (h / he)
         :ARG1 (i / it)
         :manner (c / careful))
  :op2 (s / say-01
         :ARG0 h
         :ARG1 (n / no)
         :time (t / then)))
```

Problems

Alignment Problem

1). concept not aligned



Problems

Alignment Problems

2). concept alignment error

```
# ::id lpp_1943.54 ::date 2012-07-24T11:56:15 ::annotator ISI-AMR-05 ::preferred
# ::snt The grown - ups discouraged me in my painter 's career when I was six years old , and I never
learned to draw anything , except boas from the outside and boas from the inside .
# ::zh 六岁时, 大人们使我对我的画家生涯失去了勇气, 除了画过开着肚皮和闭着肚皮的蟒蛇, 后来再没有学过画。
# ::save-date Tue Apr 9, 2013 ::file lpp_1943_54.txt
(d / discourage-01
  :ARG0 (g / grown-up)
  :ARG1 (i / i)
  :ARG2 (c / career
          :mod (p3 / paint-02
                 :ARG0 i))
  :ARG0-of (c2 / cause-01
             :ARG1 (l / learn-01
                     :ARG0 i
                     :ARG1 (d2 / draw-01
                             :ARG0 i
                             :ARG1 (a2 / anything))
                     :polarity -
                     :prep-except (a3 / and
                                    :op1 (b / boa
                                           :prep-from (o / outside))
                                    :op2 (b2 / boa
                                           :prep-from (i2 / inside)))
```

Error Statistics

Alignment:

1. total number of sentences: 1562
2. total number of words: 21381
3. number of root not aligned: 143
4. number of concept not aligned ~1253
4. number of concept alignment error ?

Solutions:

1. for unaligned node -> add action
2. for alignment error -> maybe improve the alignment

Validation

The transition system can prove to be valid.

- It visits each node and each edge with bottom-up style.
- It guarantees that each node i popped out of the σ buffer has correct children.
  - action delete edge and add child only change current node’s children
  - swap and replace head can be applied with restrictions which also does not change i’s children