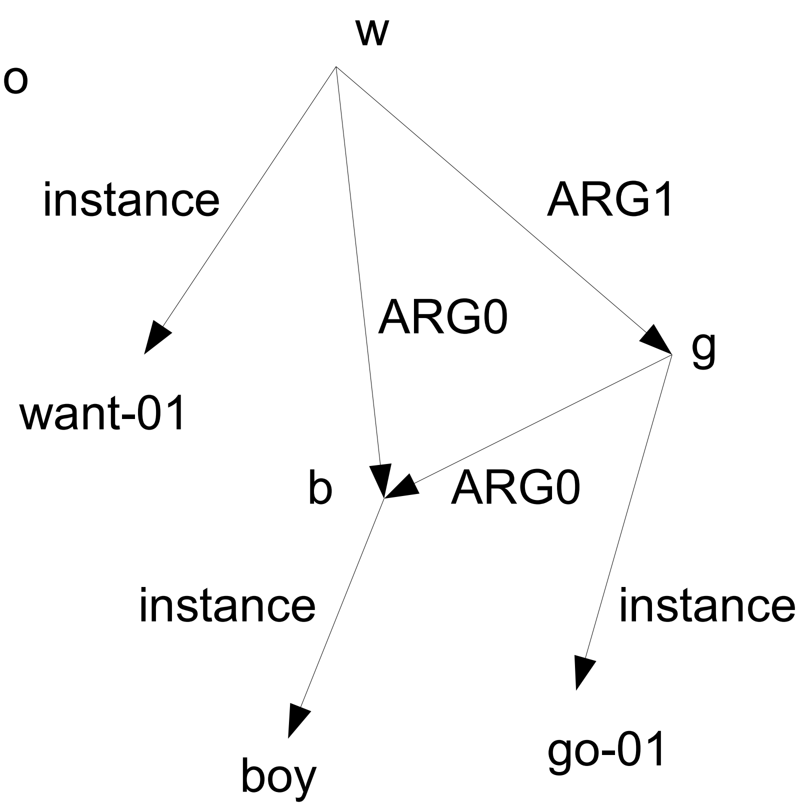
Abstract Meaning Representation (AMR)

The boy wants to go




```
(w / want-01
    :ARG0 (b / boy)
    :ARG1 (g / go-01
              :ARG0 b))
```

Abstract Meaning Representation (AMR)

Basic “who-is-doing-what-to-whom”

Cover all sentence content in single, rooted structure, DAG

Builds upon PropBank

Uses PB rolesets: e.g. describe.01

- Arg0: describer
- Arg1: thing described
- Arg2: secondary attribute, described-as

Adds more noun phrase structure, coreference and discourse structure

Abstract Meaning Representation (AMR)

AMR composed of concepts and relations, not nouns and verbs

Currently ~100 relations, plus inverses

AMR is not enslaved to syntax, or even mildly indentured:

He described her as a genius.
As he described her, she is a genius.
His description of her: a genius.

```
(d / describe-01
   :ARG0 (h / he)
   :ARG1 (s / she)
   :ARG2 (g / genius))
```

Aligning parallel corpora

Subtrees of dependency parses of parallel English/Chinese corpora only have isomorphic matches about 30% of the time.

- Yuan Ding, Thesis, 2005

Parallel PropBank structures match almost 60%.

- Wu & Palmer, SSST, 2011

What about AMR's? Will they align even more?

- Xue, Bojar, Hajič, Palmer, Urešová, Zhang, LREC 2014

***M**eaning in **A**MR's and **T**ectogrammatical **R**epresentation **Interchange** (MATRIX)*

Martha Palmer
(Colorado)

Jan Hajic
(Charles)

Nianwen Xue
(Brandeis)

- Zdenka Uresova (Charles)
- Ondrej Dusek (Charles)
- Tim O'Gorman (Colorado)
- Ondrej Bojar (Charles)

MATRIX Questions

***M**eaning in **A**MR's and **T**ectogrammatical **R**epresentation **Interchang**e*

How distant/similar are AMR's and the Tectogrammatical Representation for English? Can we port the TR MT system to AMR's?

How distant/similar are English AMR's, Chinese, and Czech AMR's?

Which differences have the most impact on the graph matching?

How much can deterministic reformatting of AMR's bridge the distances?

Preparatory Efforts

English, Chinese, and Czech AMR’s of the same 100 sentences and their translations.

A preliminary mapping from TR to AMR.

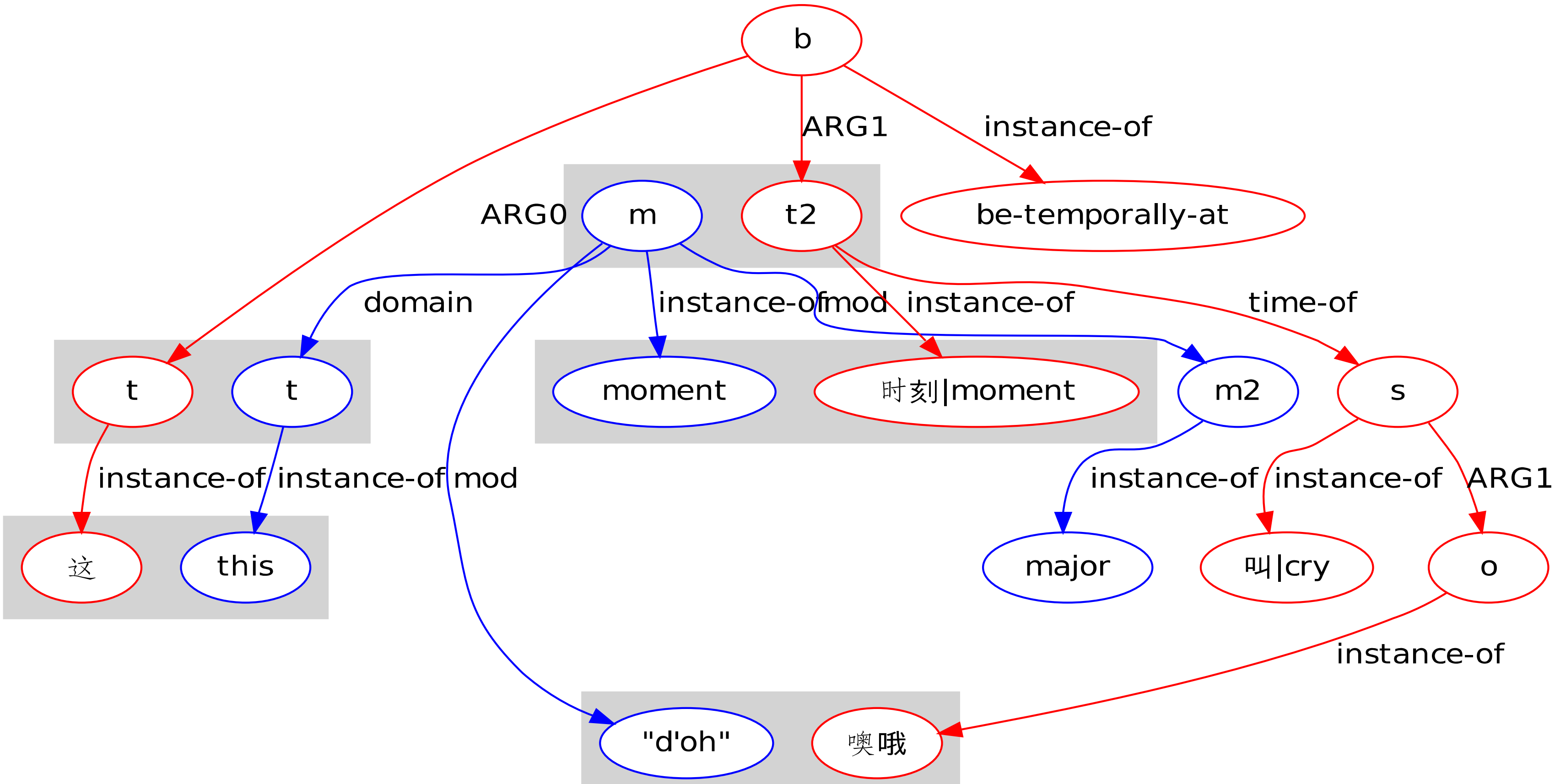
Given a 1M word WSJ English corpus with parallel Czech translations, both in TR

And automatically produced AMR’s (from OntoNotes, thanks to Ulf Hermjakob) for the same data

Differences in Lexicalization and Annotation Choice

这 是 一 个 大 叫 “ 噢哦 ！ ” 的 时刻 。

This is a major `` D'oh! '' moment .



Annotation Choice Differences

Annotation choice

To reify or not to reify?

Chinese: reifies “be_temporally_located_at”

English drops “be” and puts “this” as the :domain of “moment”:

(m / moment

- :mod (m2 / major)
- :domain (t / this)
- :mod (d / d'oh :mode expressive))

Alternatives Annotation Choices for English

English could just as easily reify “is moment” as ***temporal_location.01***

(t / temporal_location.01

- :Arg1 (t2 / this)
- :mod (m / major)
- :mod (d / d'oh :mode expressive))

English and Chinese would match more closely

How often is this the case?

Lexicalization differences

Language specific lexicalization differences

- Simply different word choices
  - “major” vs. **叫 /** cry

Often a single lexical item in one language is a multi-word expression elsewhere, w/ structure

“tells the tale” vs. popsány..

- (t / tell.01 (p / popsat.1

:Arg1 (t2 / tale) (no :Arg1)

“*překračovat povolenou rychlost*” vs. “speeding”

Should AMR make more of an effort to treat MWE’s as single lexical items?

“Zácpa kolem čeho“
“Localized congestion around what”

Questions to investigate

If there are alternative annotation choices, can we deterministically produce them, resulting in better matches?

Where there are language-specific different lexicalizations, are there resources that could provide bi-lingual mappings?

How much should AMR abstract away from Multi-word expressions?

When to reify? And when not?

Etc.,

Graph-Based Parser for the Abstract Meaning Representation (JAMR)

Jeff Flanigan (CMU)

Flanigan, Thomson, Carbonell, Dyer, Smith (ACL 2014)

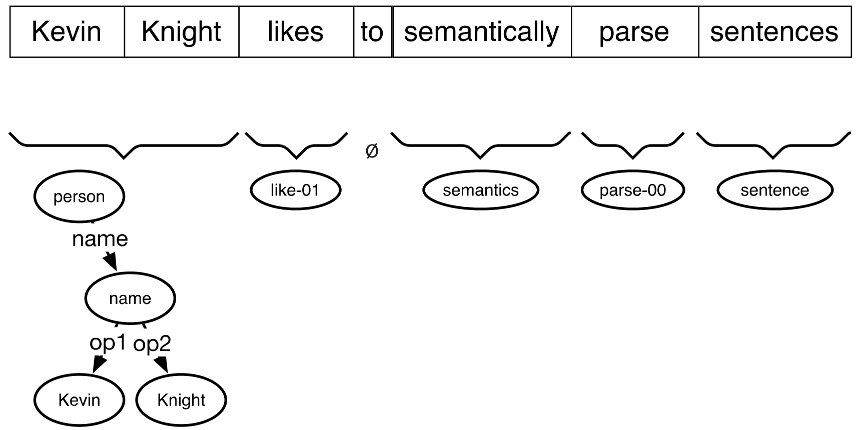
JAMR Overview

Input

Kevin	Knight	likes	to	semantically	parse	sentences

JAMR Overview

Input



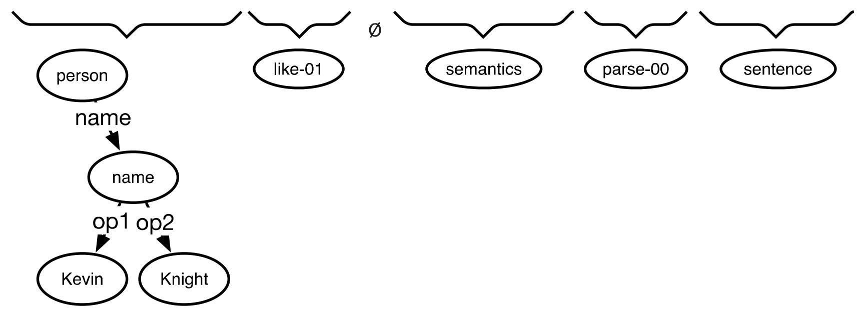
Concept Identification

- Label sentence with concept fragments

JAMR Overview

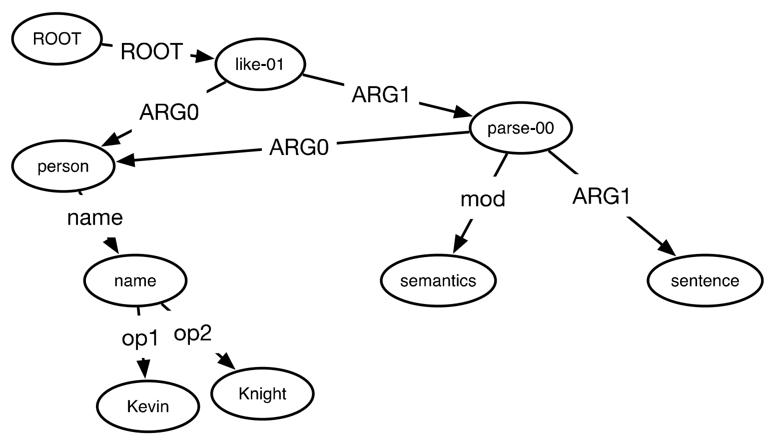
Input

Kevin	Knight	likes	to	semantically	parse	sentences



Concept Identification

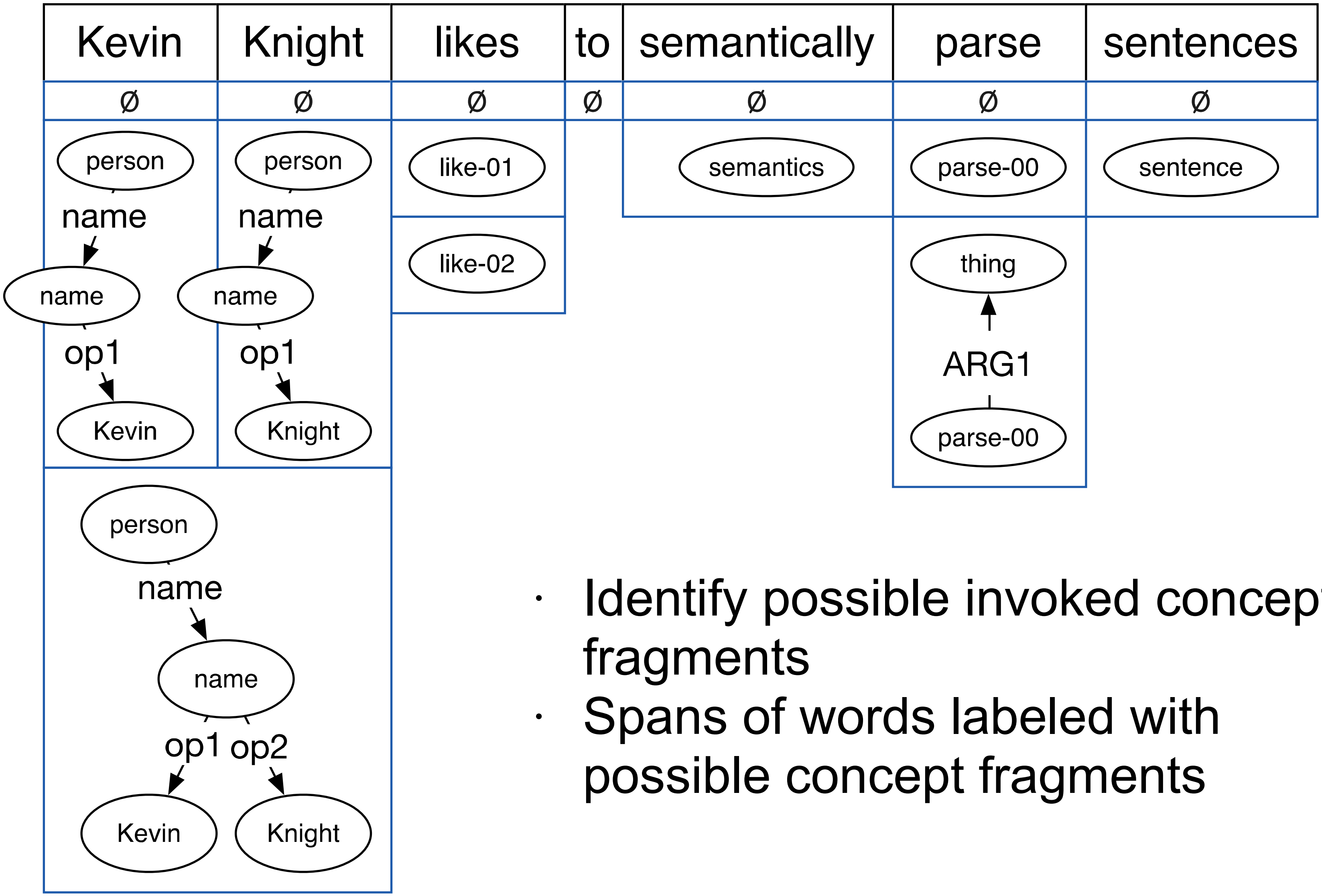
- Label sentence with concept fragments



Relation Identification

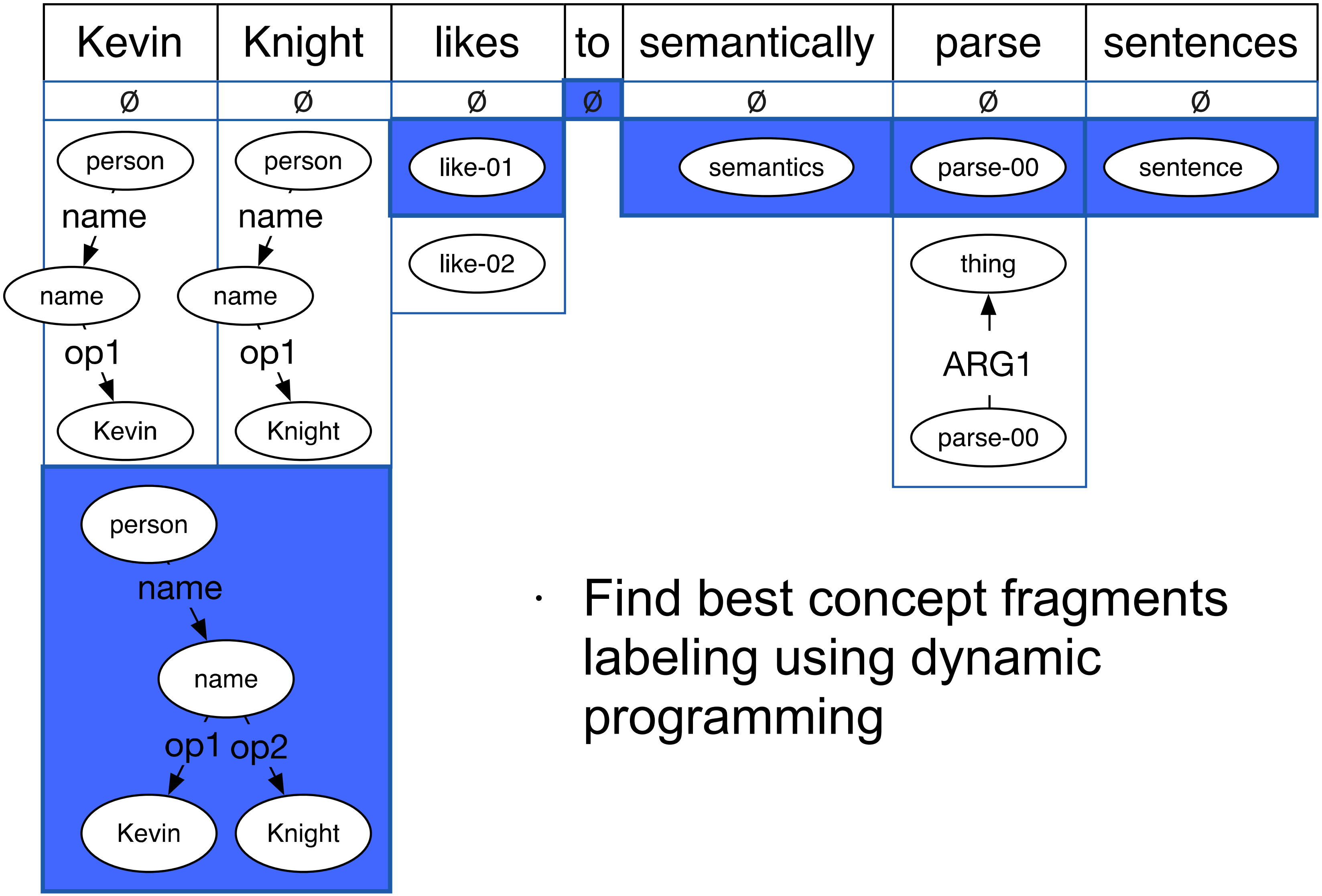
- Connect fragments by adding relations
- Relations = labeled directed edges

Concept Identification



- Identify possible invoked concept fragments
- Spans of words labeled with possible concept fragments

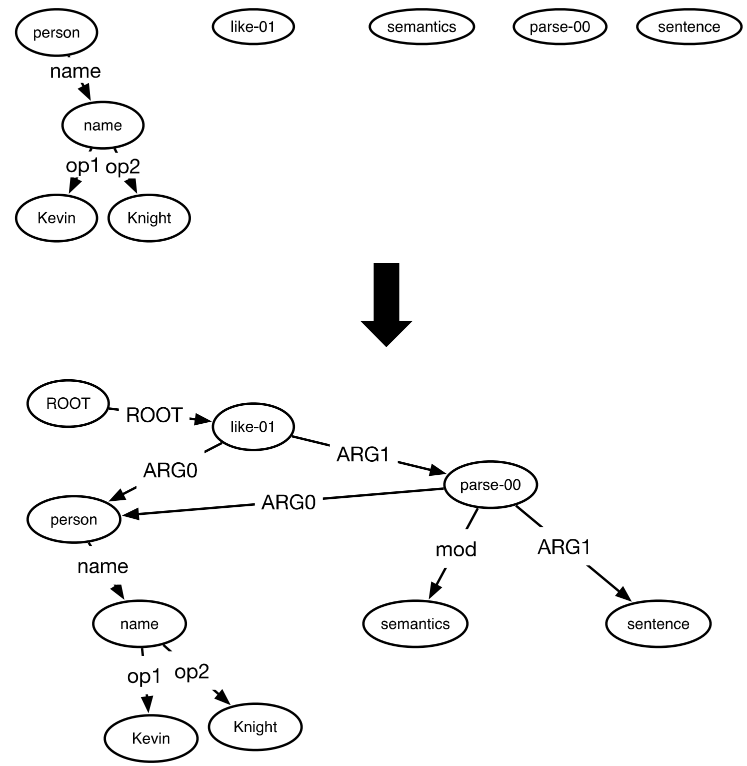
Concept Identification



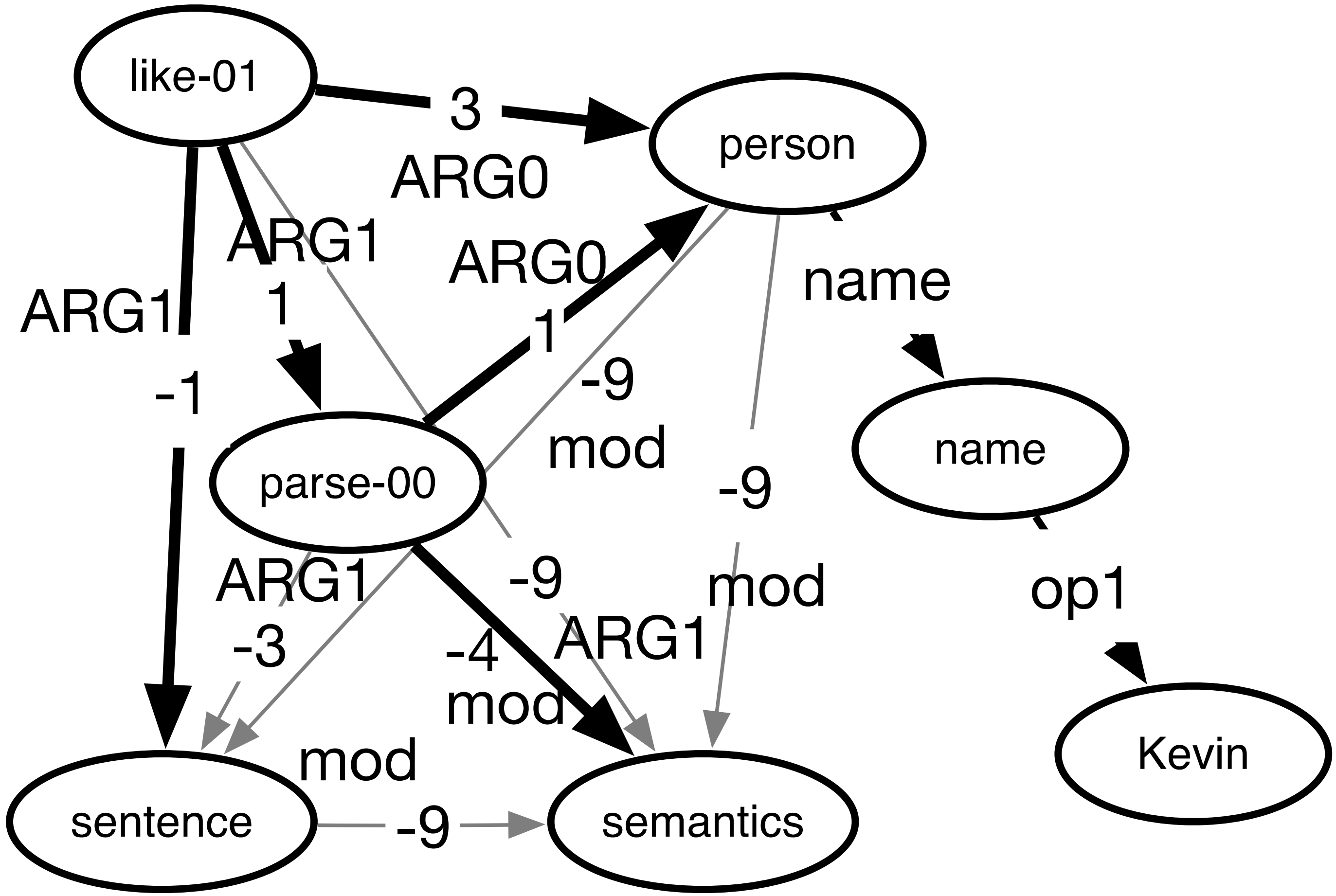
- Find best concept fragments labeling using dynamic programming

Relation Identification

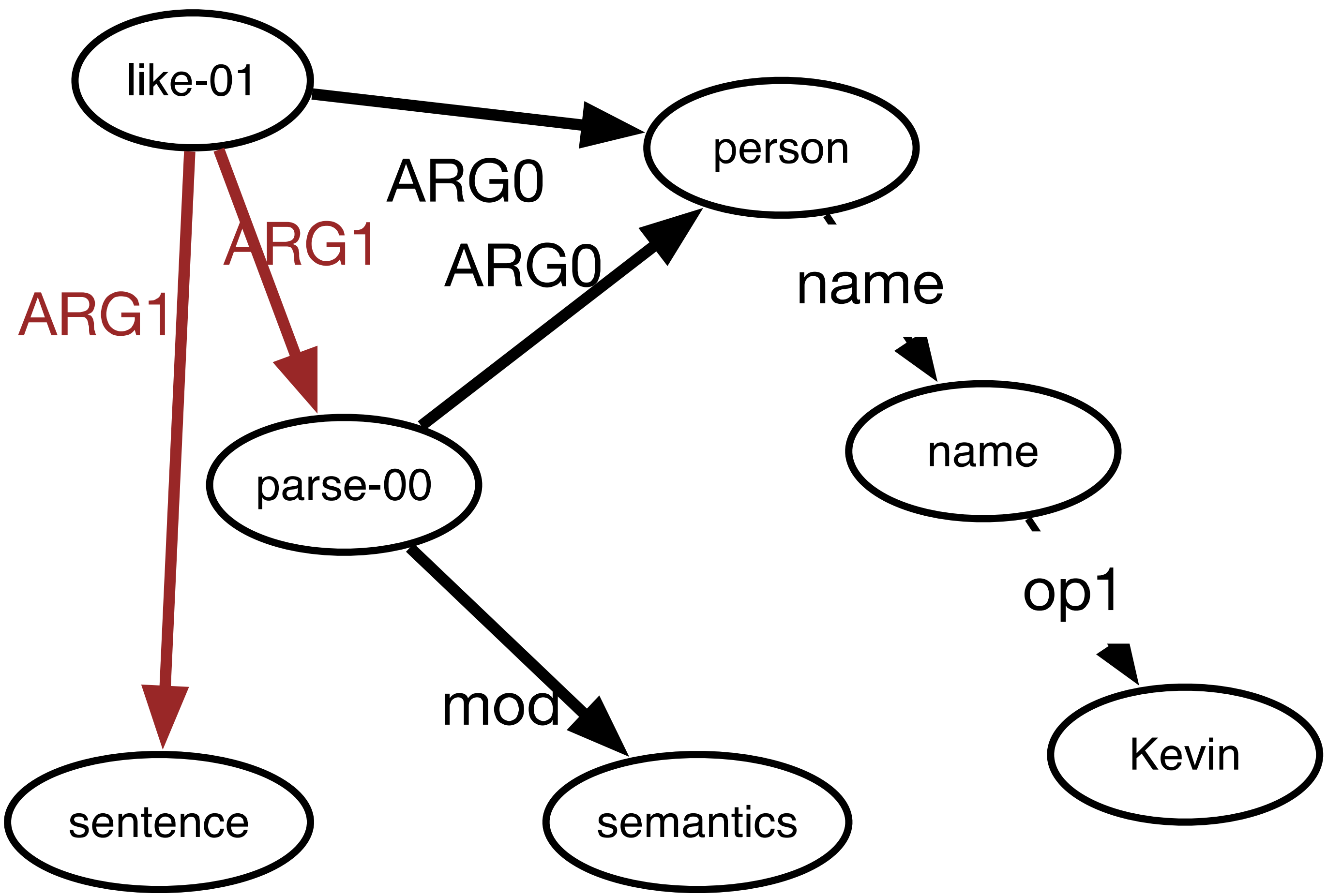
- Relation identification adds edges between nodes
- Edge factored discriminatively trained model
- Uses maximum spanning, connected, sub-graph algorithm (MSCG) with additional constraints



Maximum Spanning Connected Sub-Graph Algorithm (MSCG)

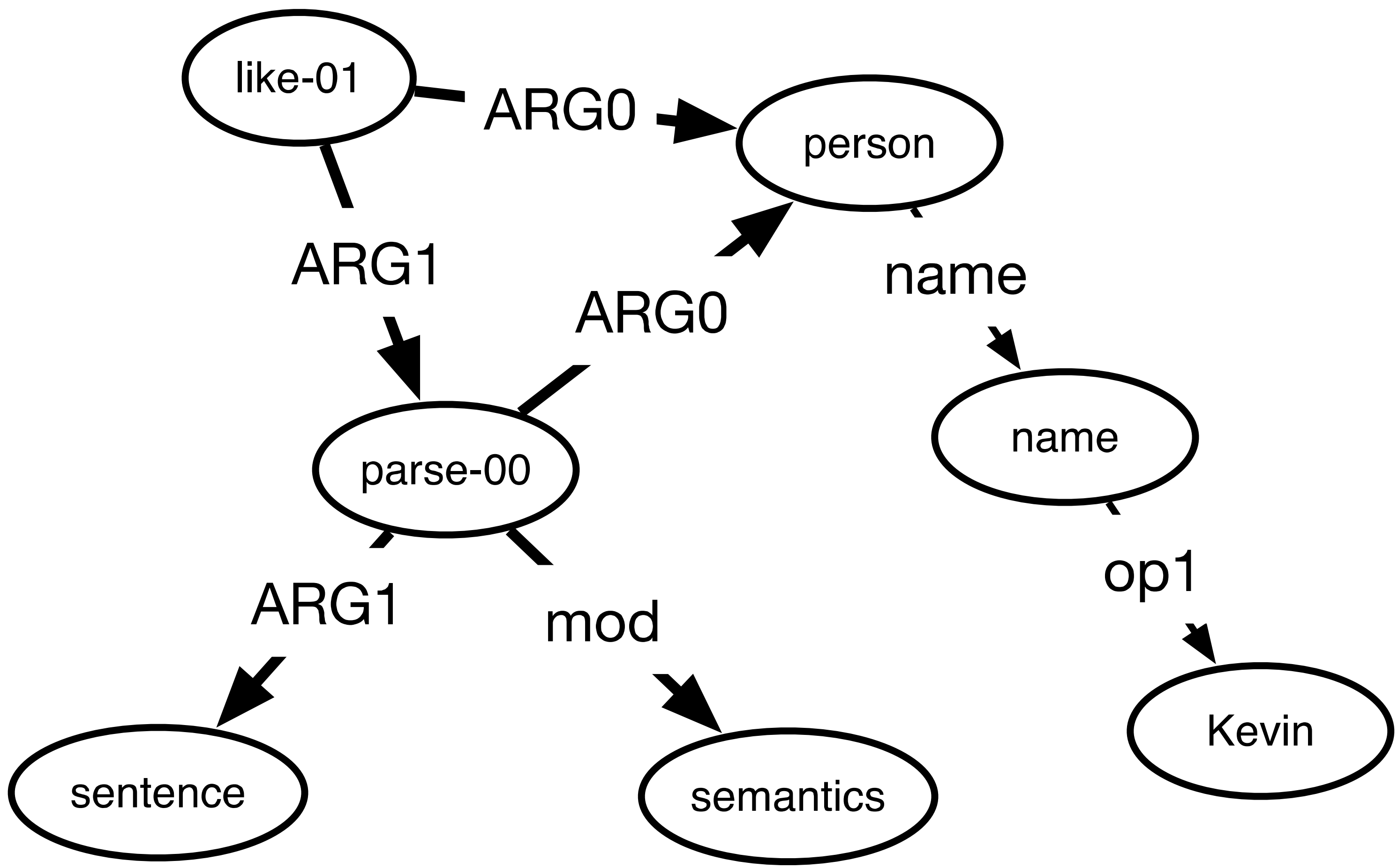


Maximum Spanning Connected Sub-Graph Algorithm (MSCG)



Graph must be deterministic

Enforce Constraint Using Lagrangian Relaxation



Results

	F1
Concept Identification	76%
Full System (gold concepts)	80% Smatch
Full System	58% Smatch

JAMR available at
http://github.com/jflanigan/jamr

Chinese AMR parsing

Nianwen Xue (Brandeis)

- Chuan Wang (Brandeis)
- Yuchen Zhang (Brandeis)
- Wei-Te (Colorado)

- Data availability:
    - Only have AMR annotation for 100 sentences. Specifications still under development
  - But there is treebank and propbank annotation for over 1.5M words
- So during the workshop we will be working on:
  - Producing pseudo AMRs based on the Chinese TreeBank and PropBank
  - Developing annotation specifications for Chinese AMRs
  - Developing a dependency tree to graph transition system, initially trained on English AMRs

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

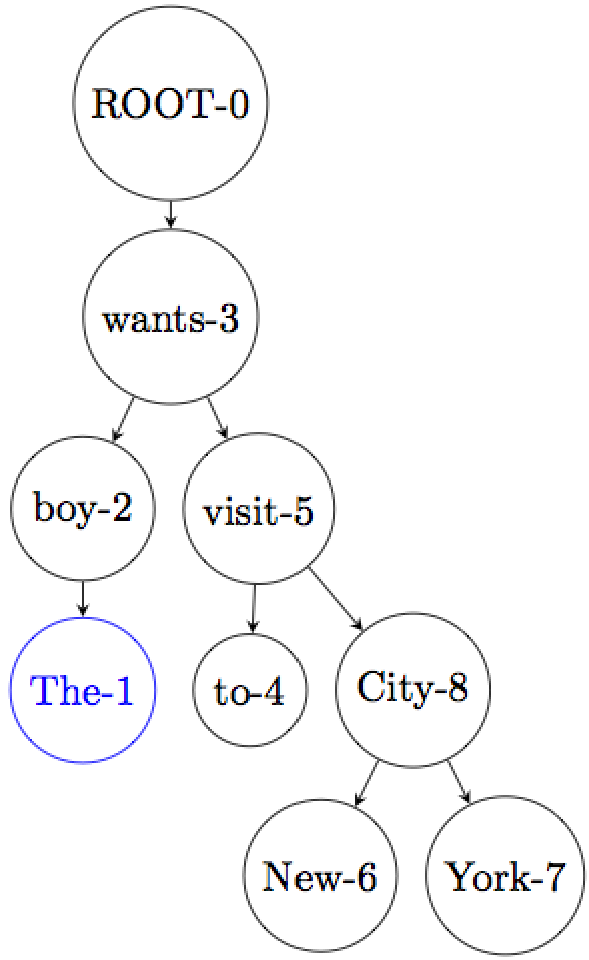
➔ if β is not empty:

- 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

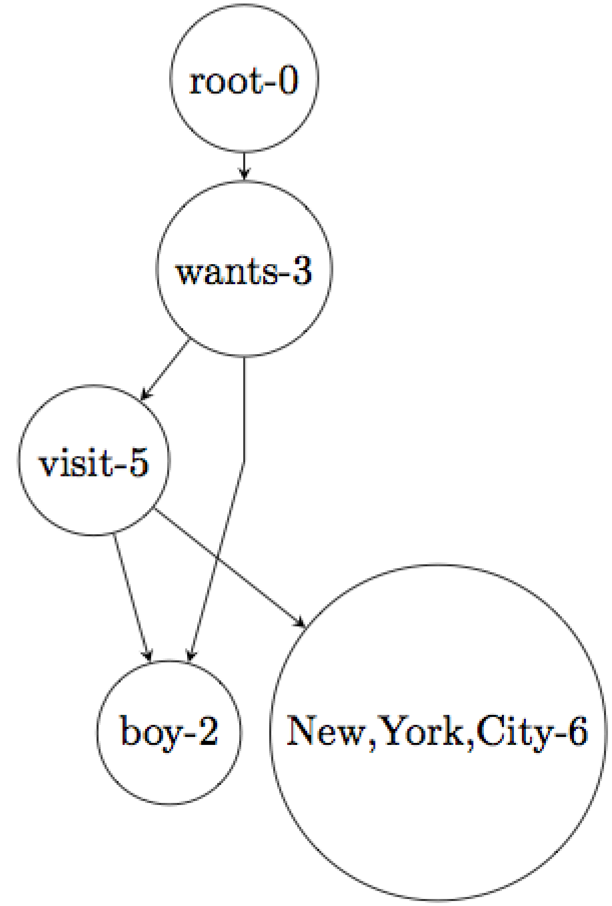
➔ β is empty:

- add child k: (i|σ,[],A) => (i|σ,[],A.add_edge(i,k))
- finish: (i|σ,[],A) => (t|σ,βt,A) # done with current node

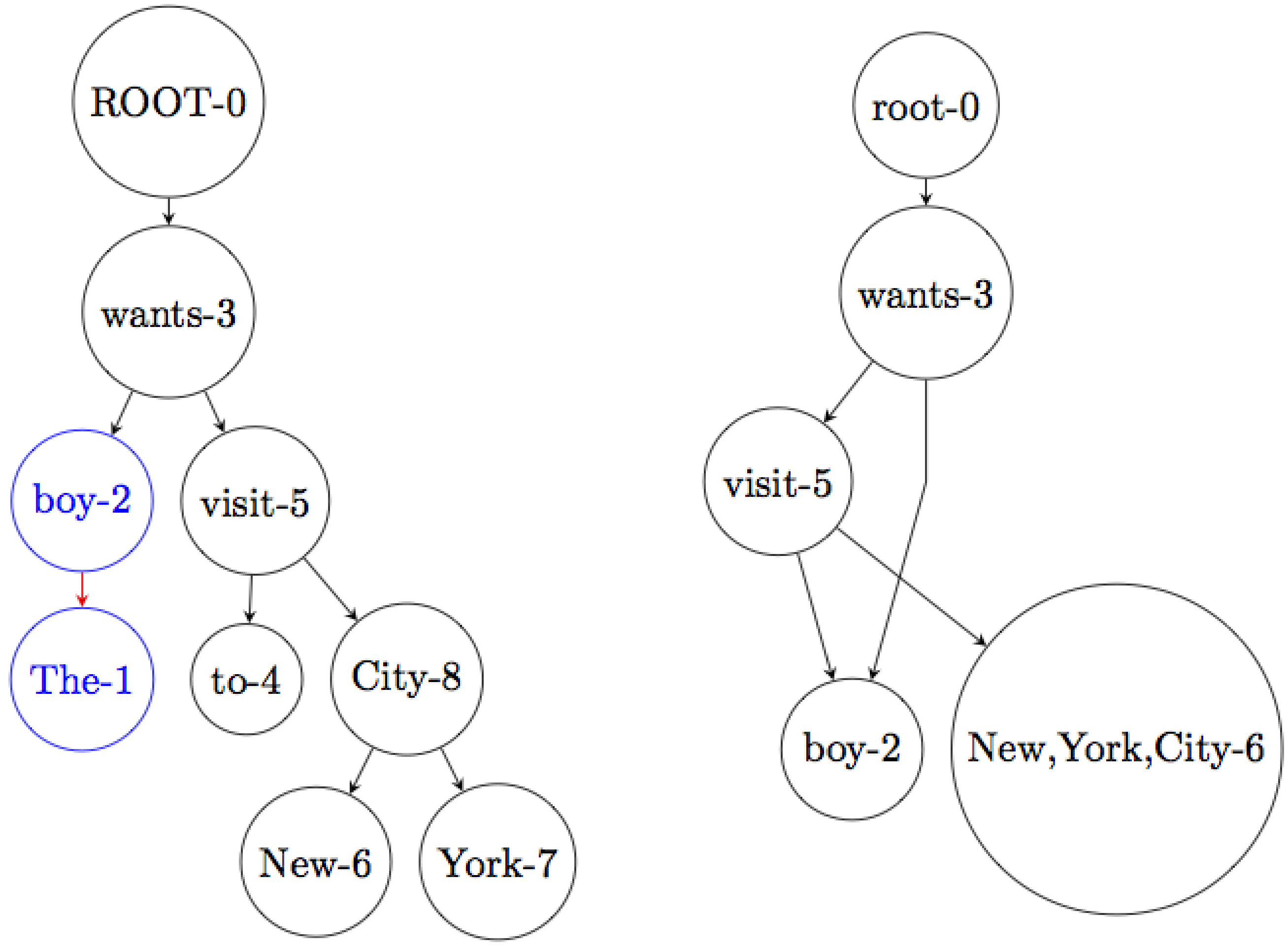
Example



σ:[The-1|σ]
β:[]
Finish



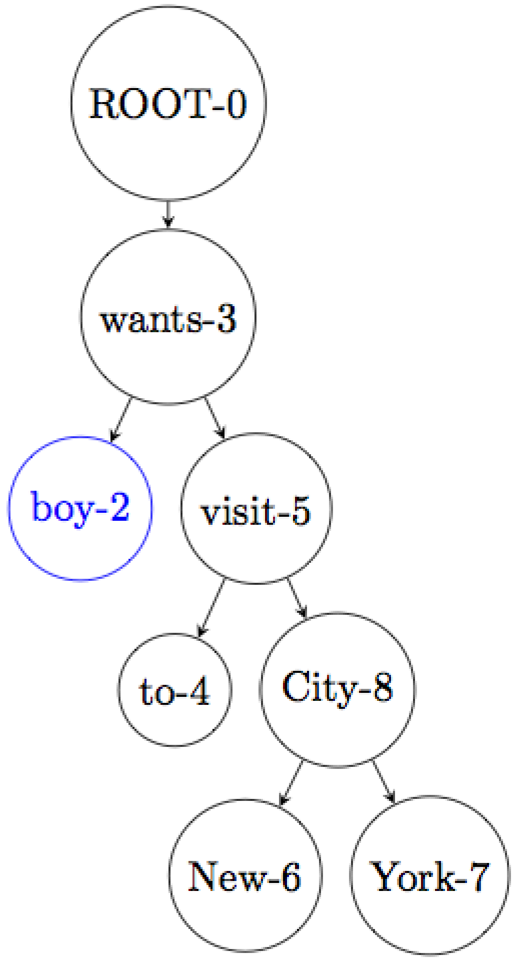
Example

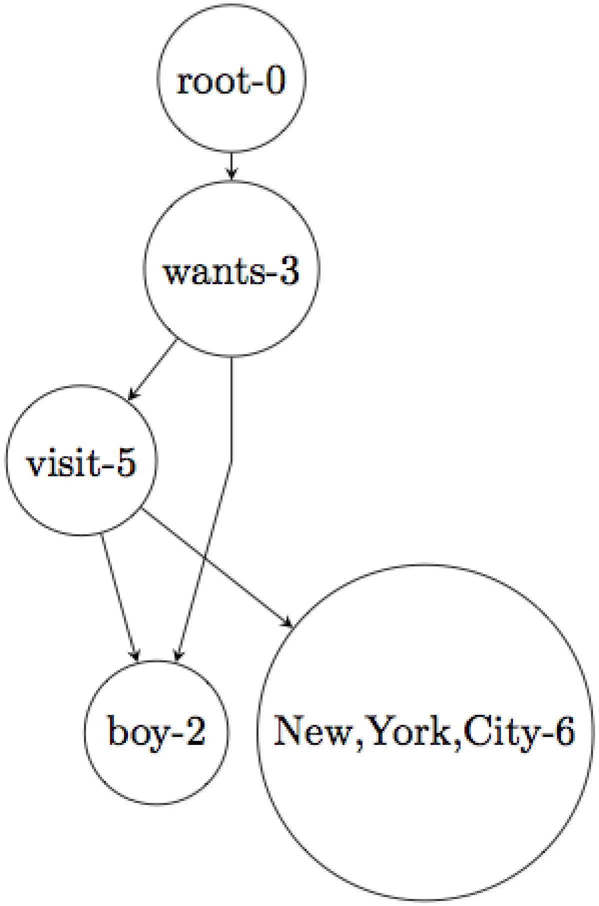


σ:[boy-2|σ]
β:[The-1]
Delete child

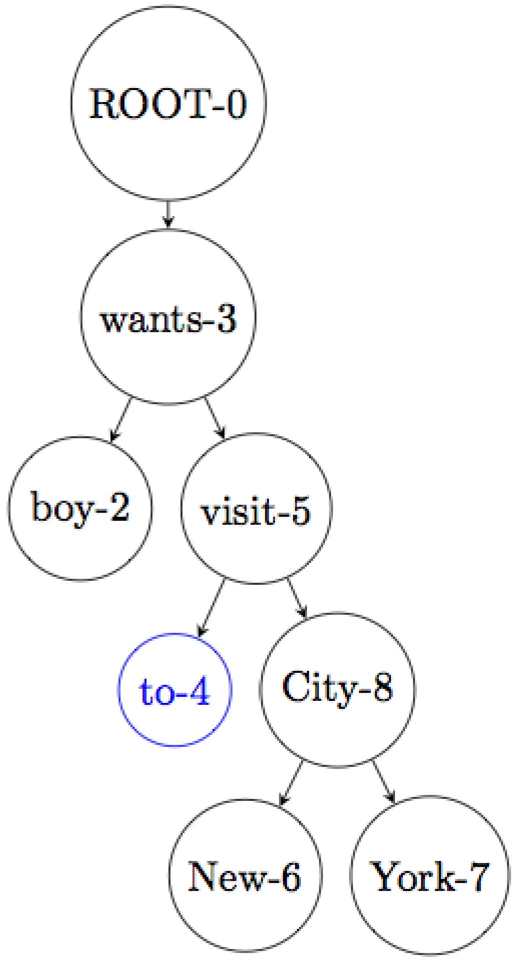
Example

σ:[boy-2|σ]
β:[]
Finish

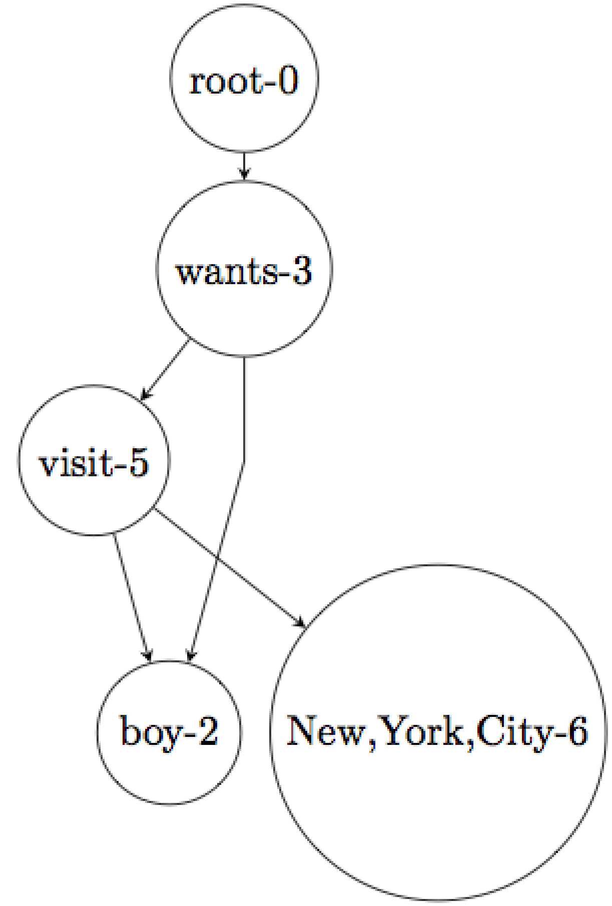




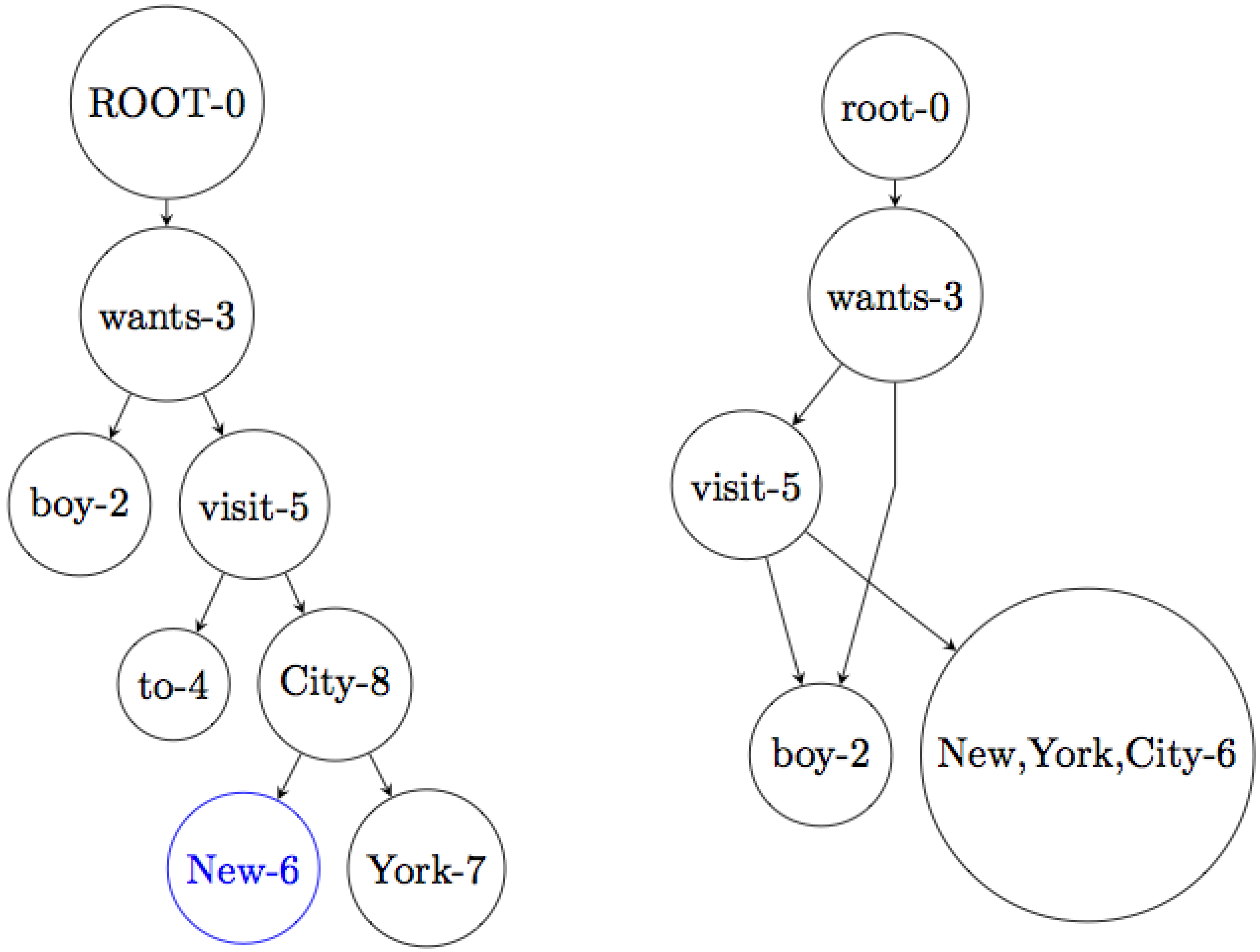
Example



σ:[to-4|σ]
β:[]
Finish



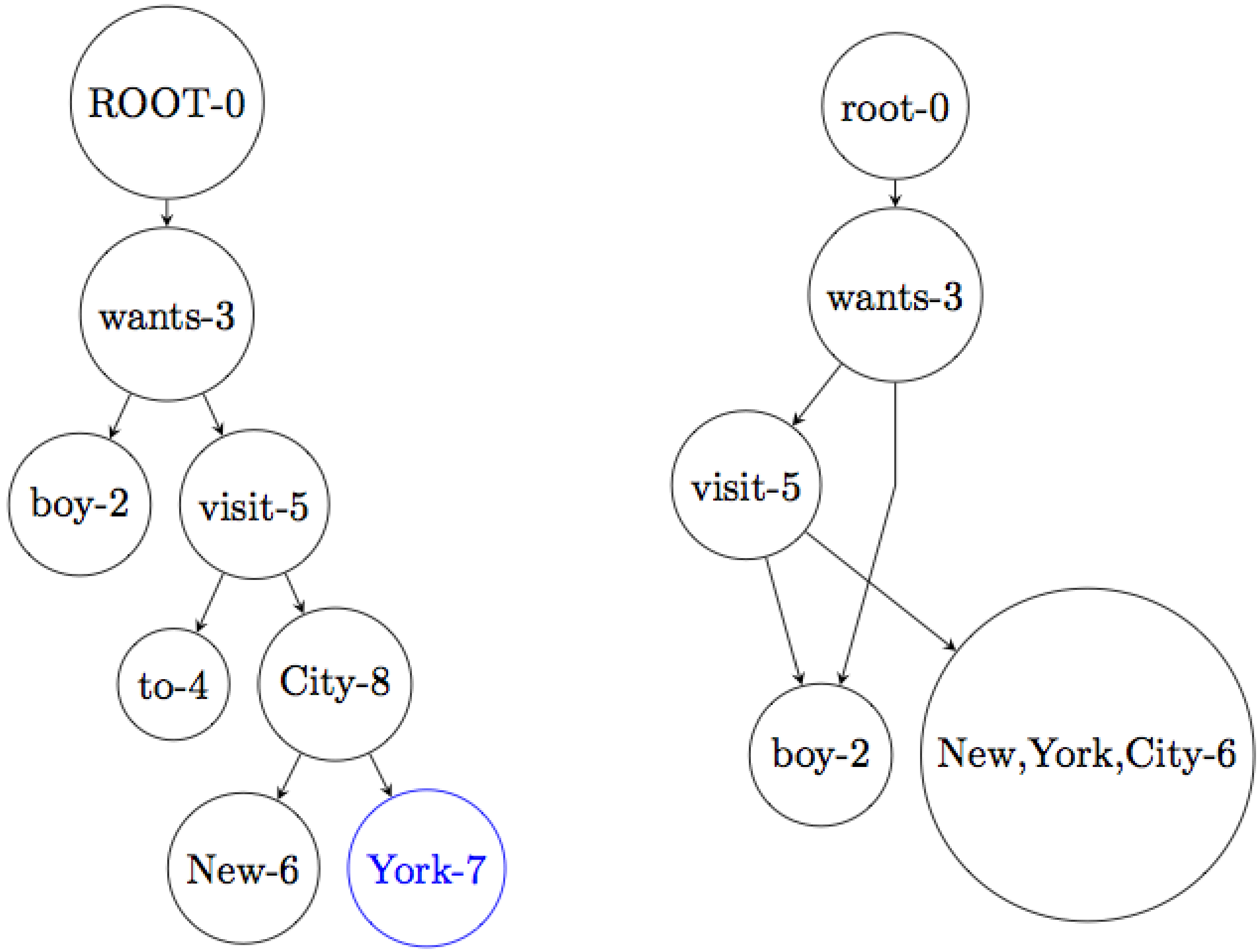
Example



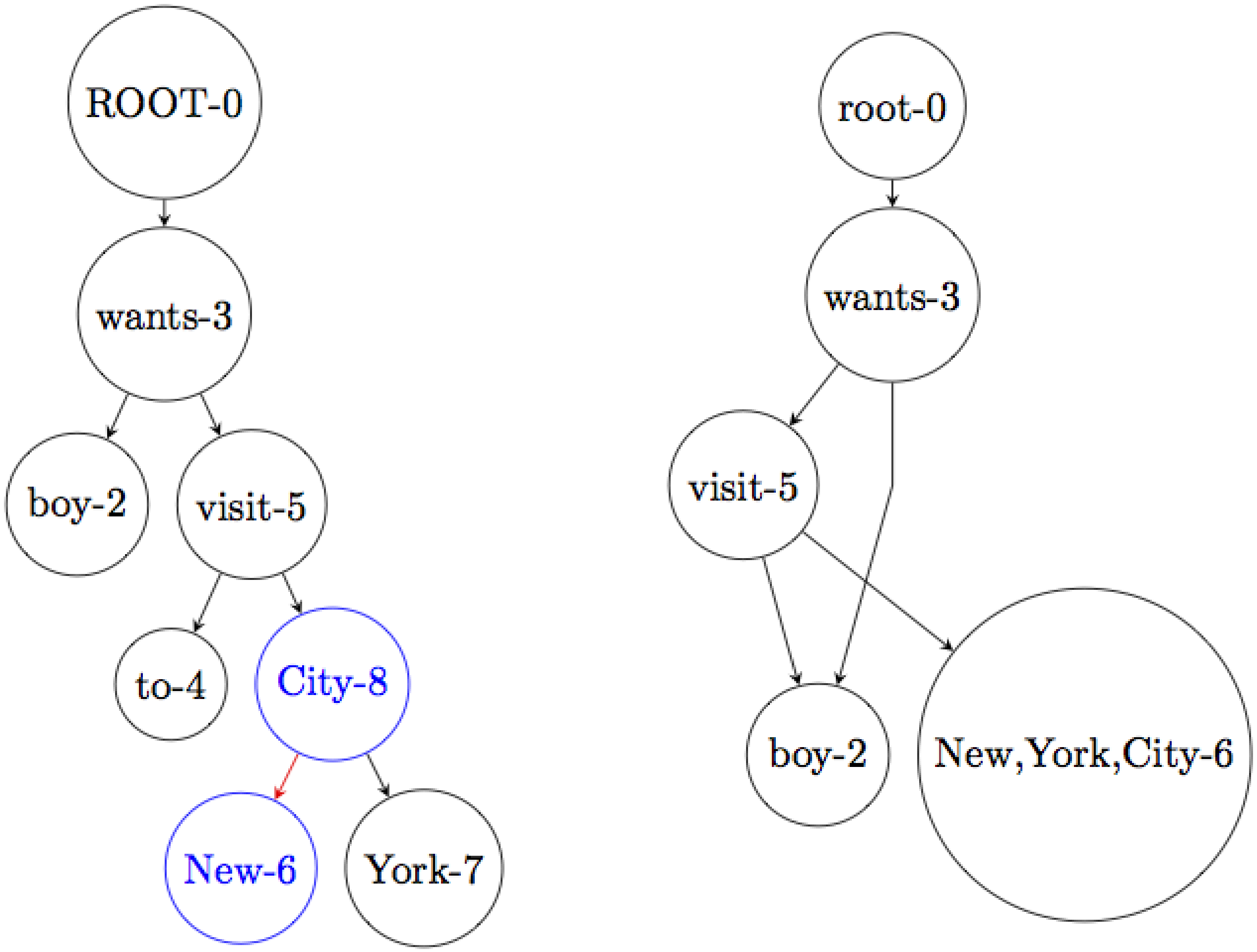
σ:[New-6|σ]
β:[]
Finish

Example

σ:[York-7|σ]
β:[]
Finish



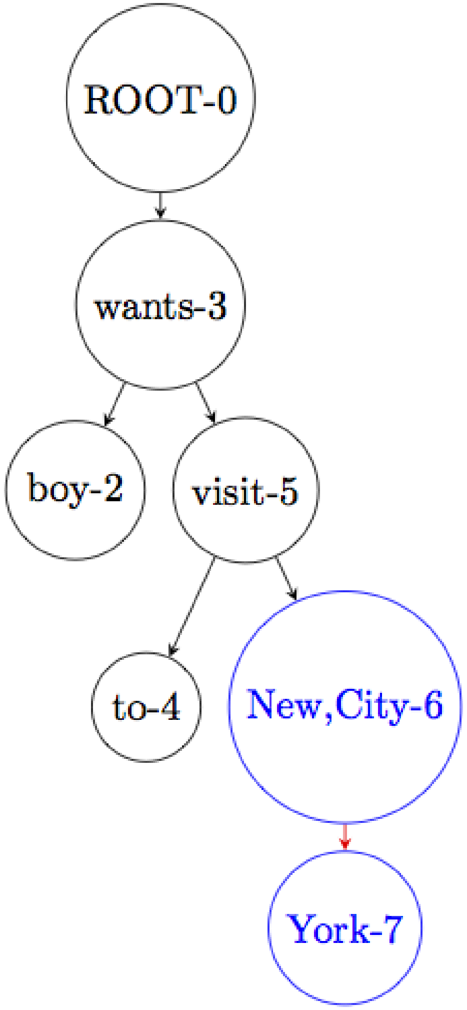
Example

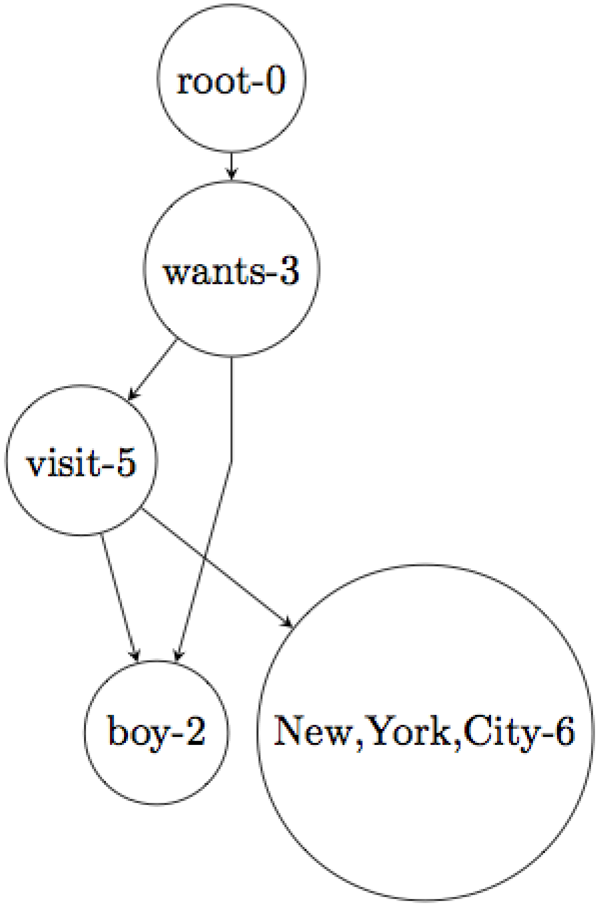


σ:[City-8|σ]
β:[New-6|β]
Merge

Example

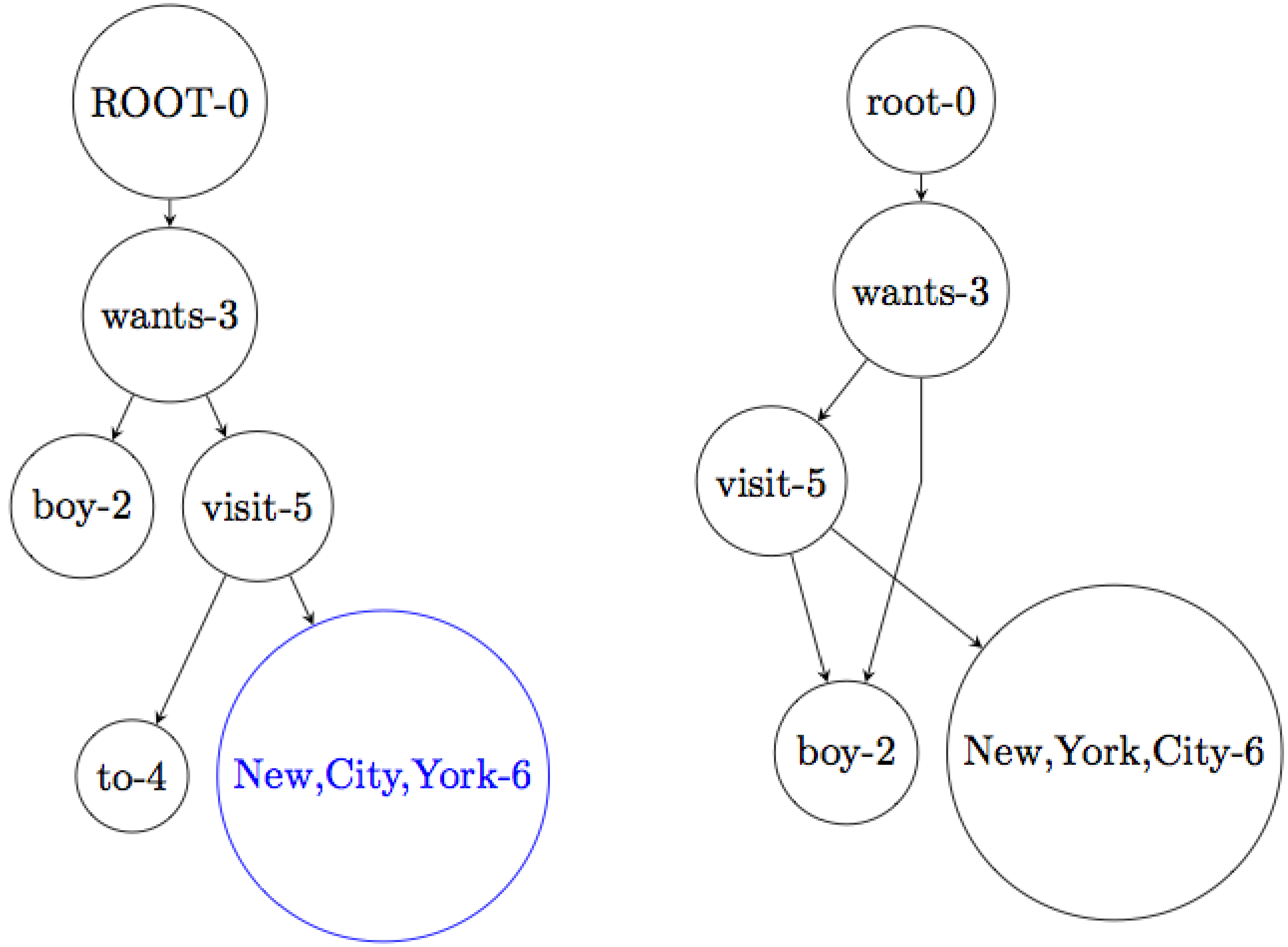
σ:[New,City-6|σ]
β:[York-7]
Merge



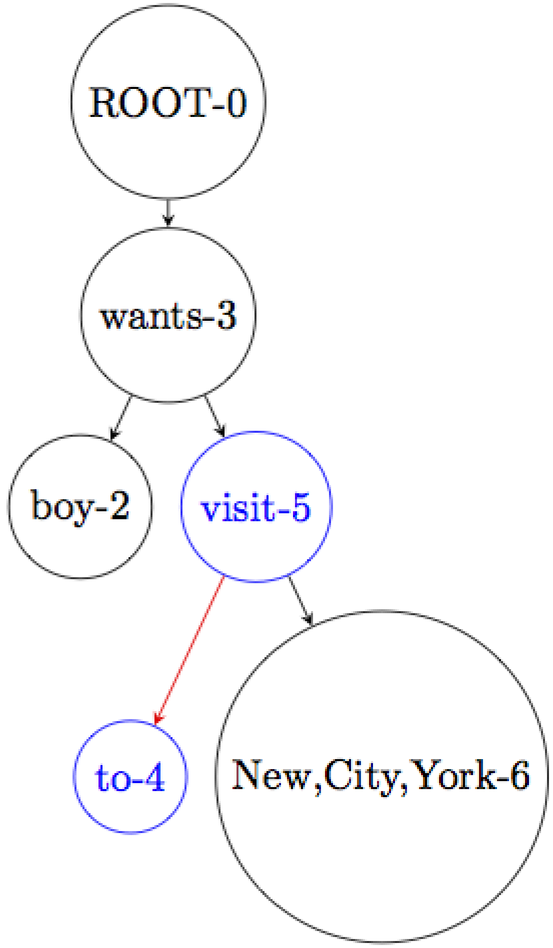


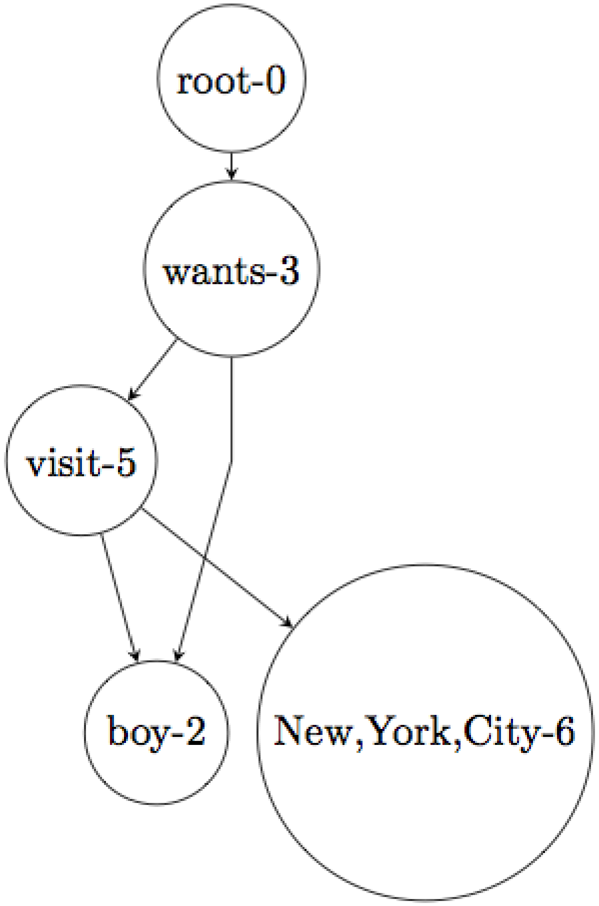
Example

σ:
[New,York,City-6|σ]
β:[]
Finish



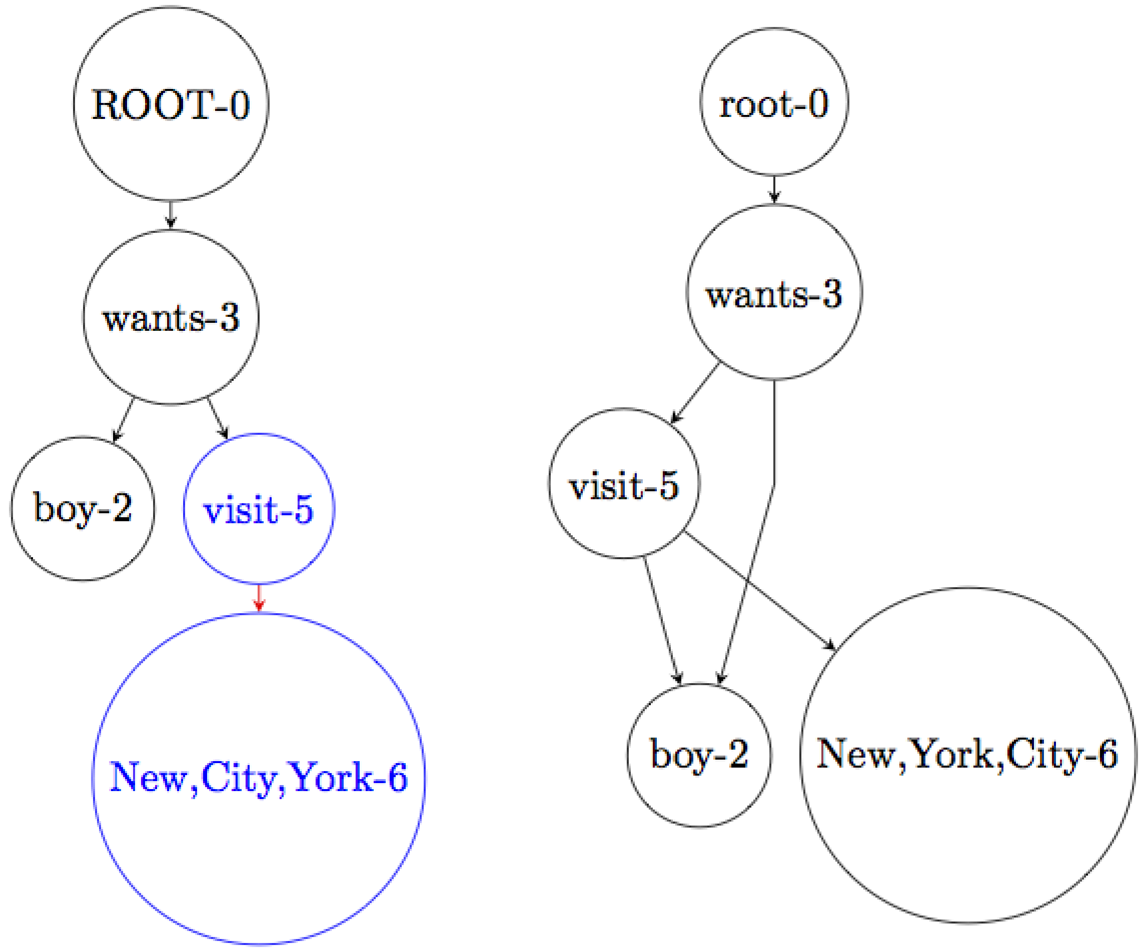
Example





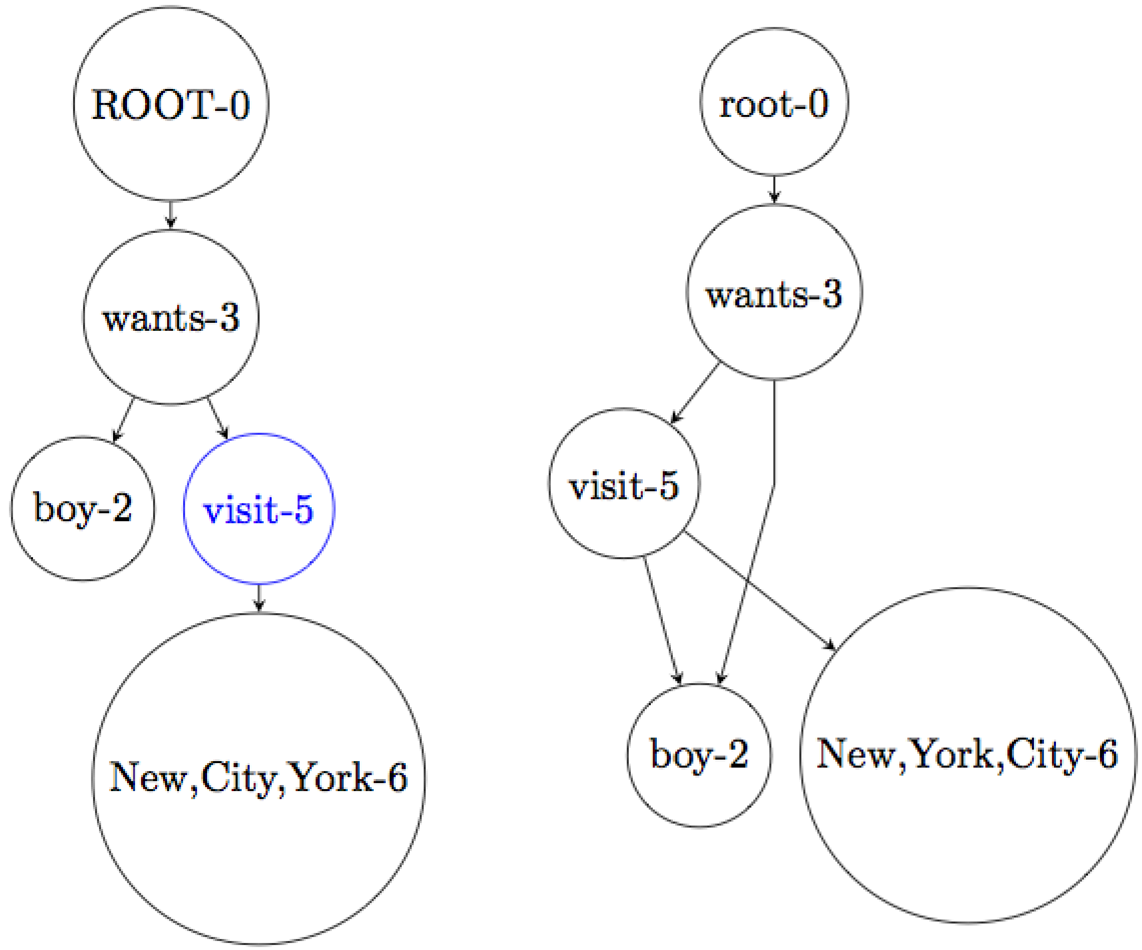
σ:[visit-5|σ]
β:[to-4|β]
Delete child

Example



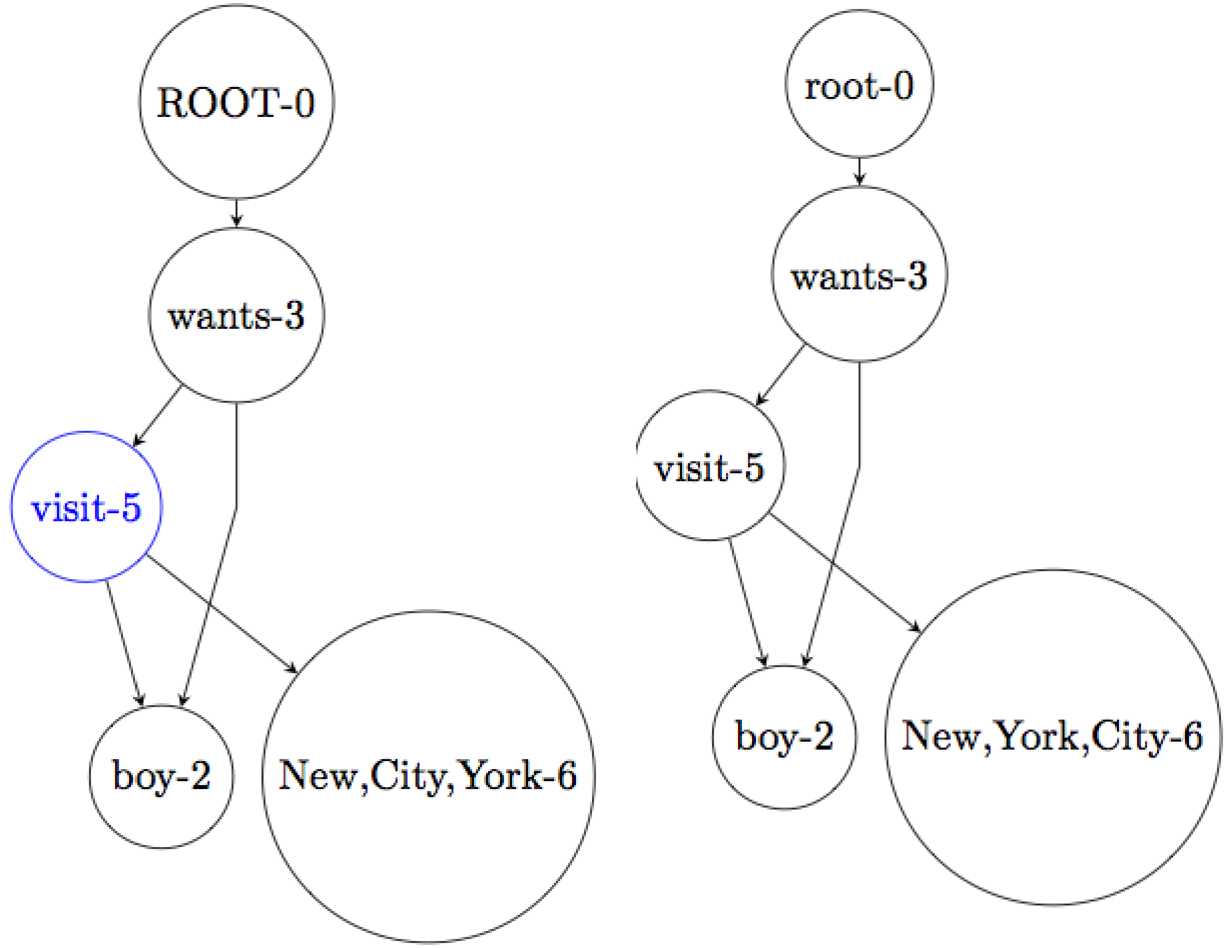
σ:[visit-5|σ]
β:
[New,York,City-6]
Next

Example



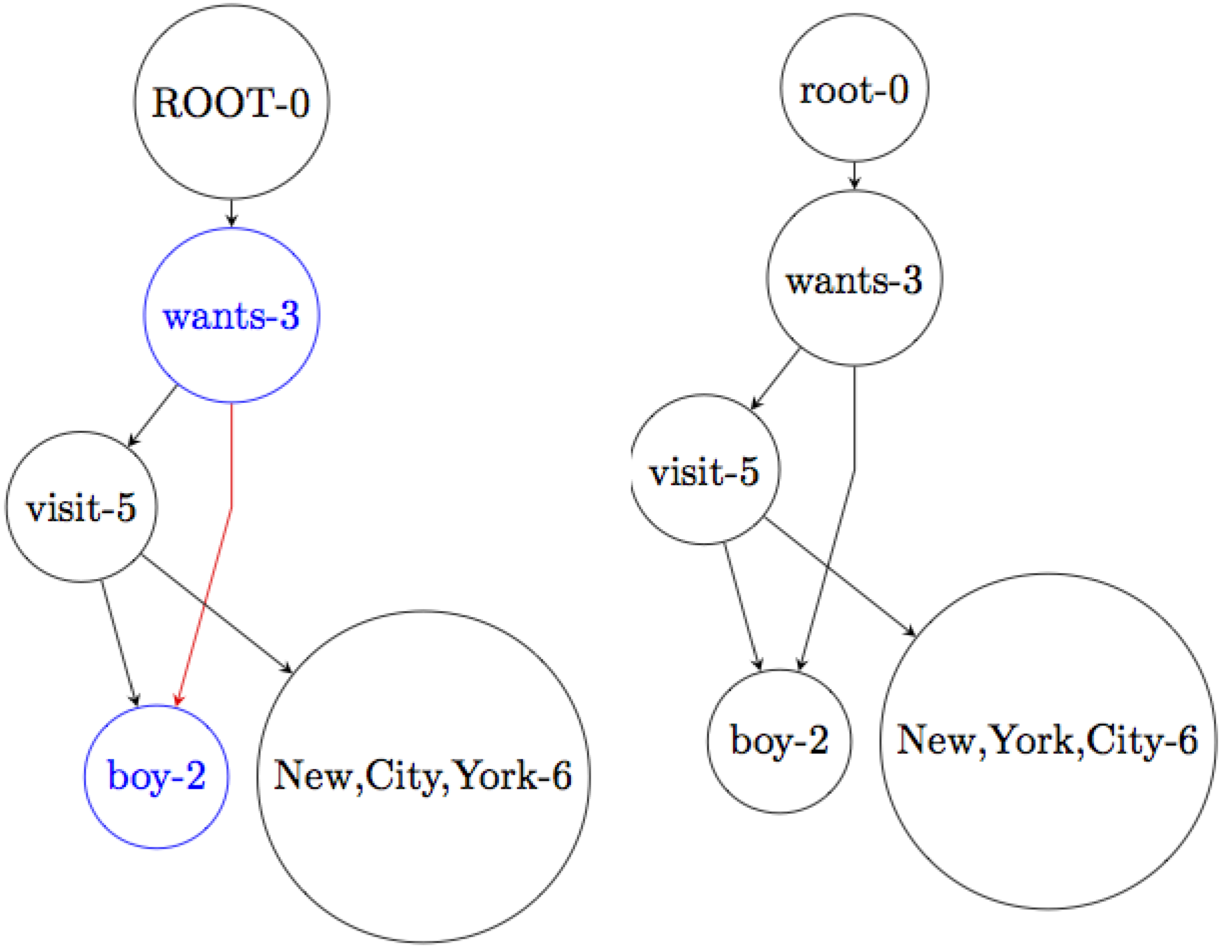
σ:[visit-5|σ]
β:[]
Add child boy-2

Example



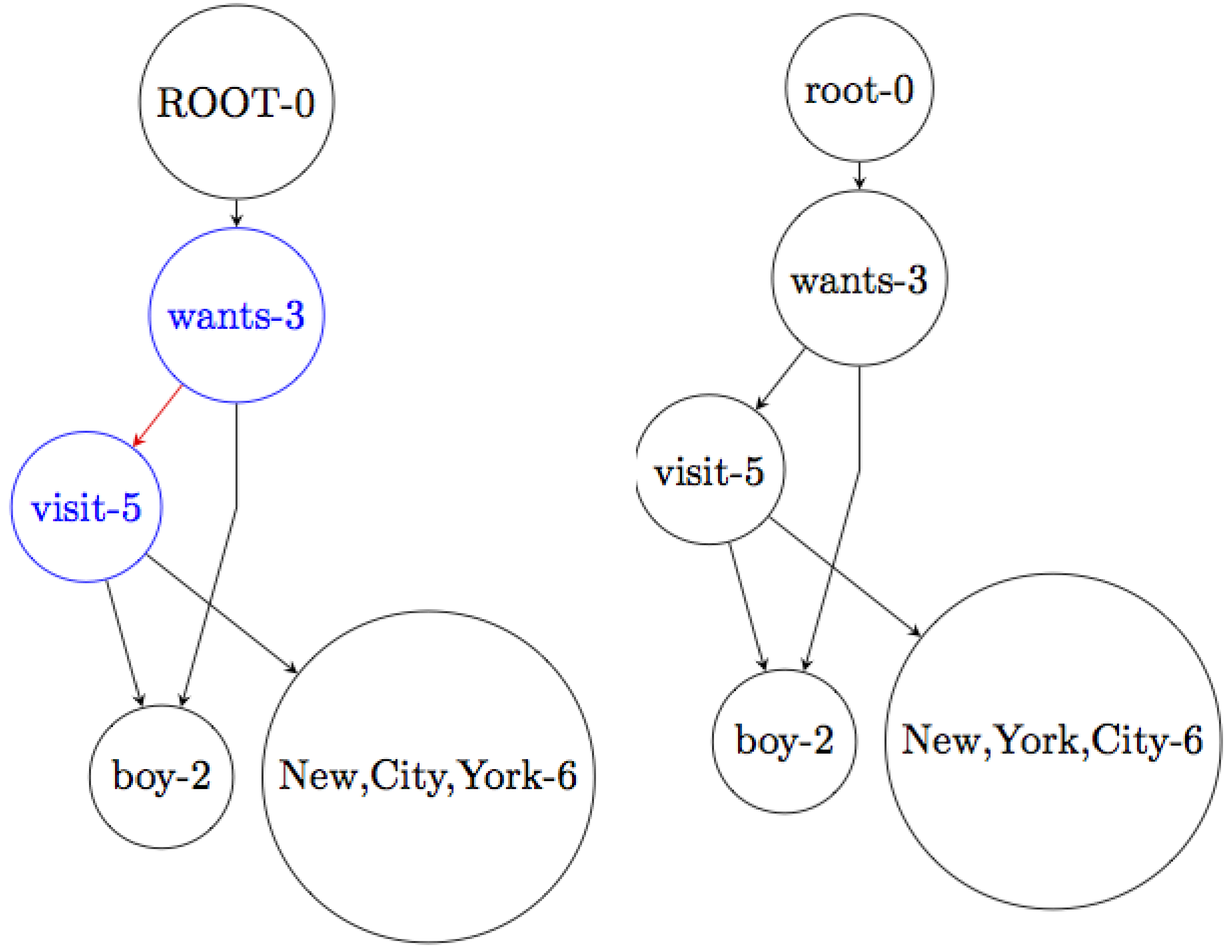
σ:[visit-5|σ]
β:[]
Finish

Example



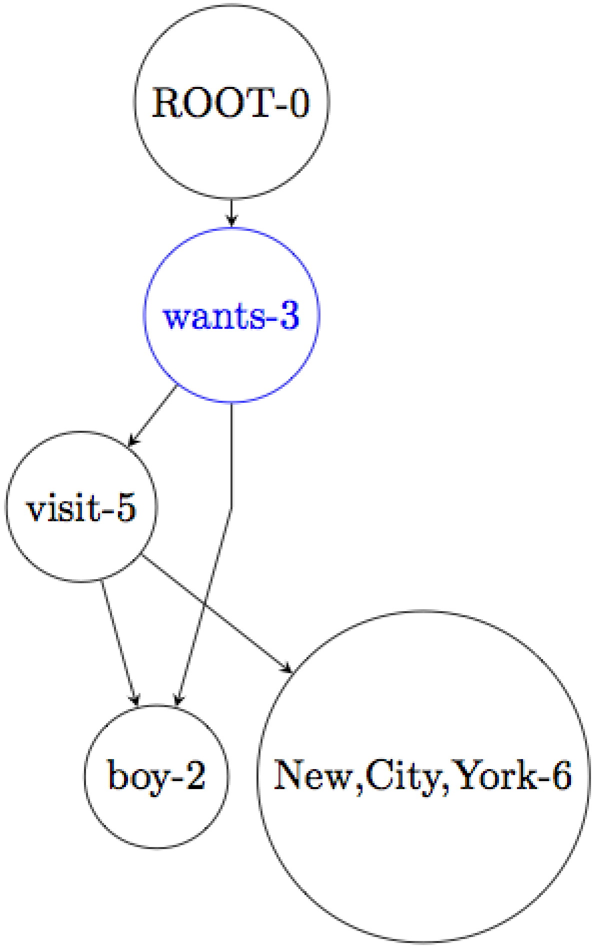
σ:[wants-3|σ]
β:[boy-2|β]
Next

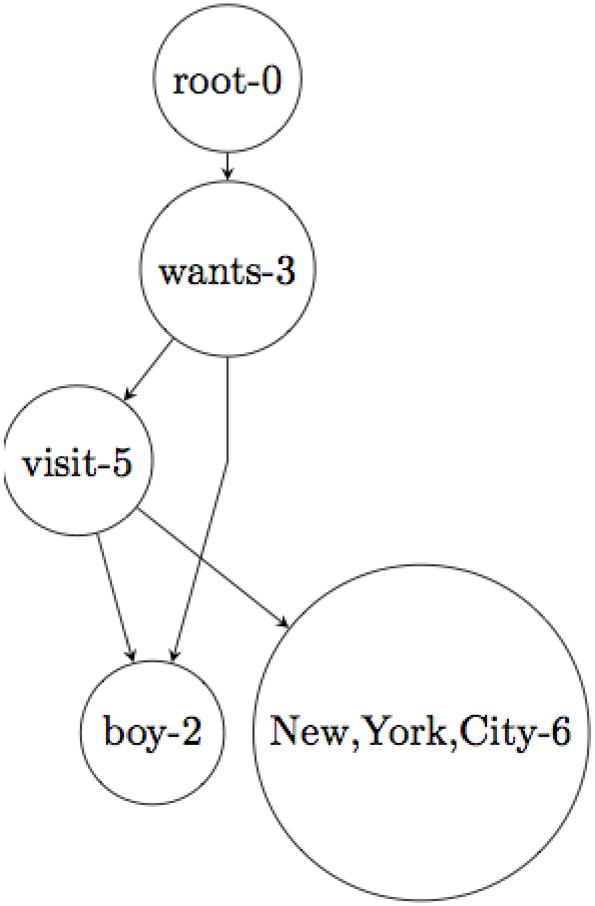
Example



σ:[wants-3|σ]
β:[visit-5]
Next

Example

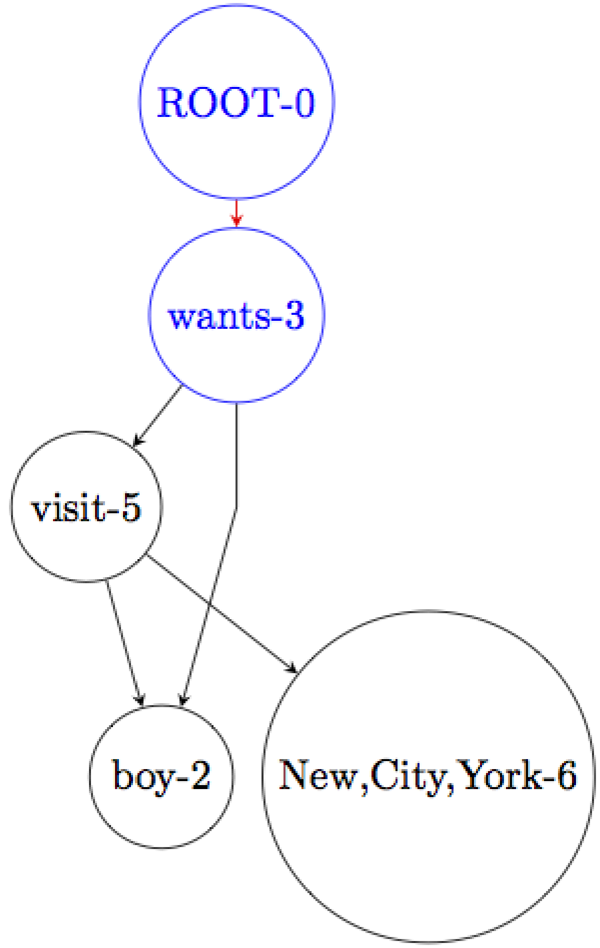


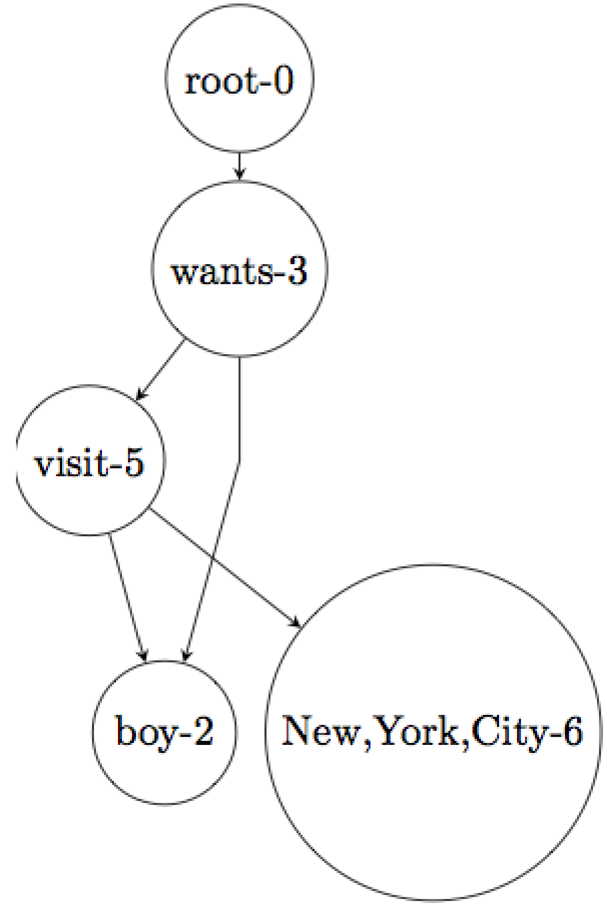


σ:[wants-3|σ]
β:[]
Finish

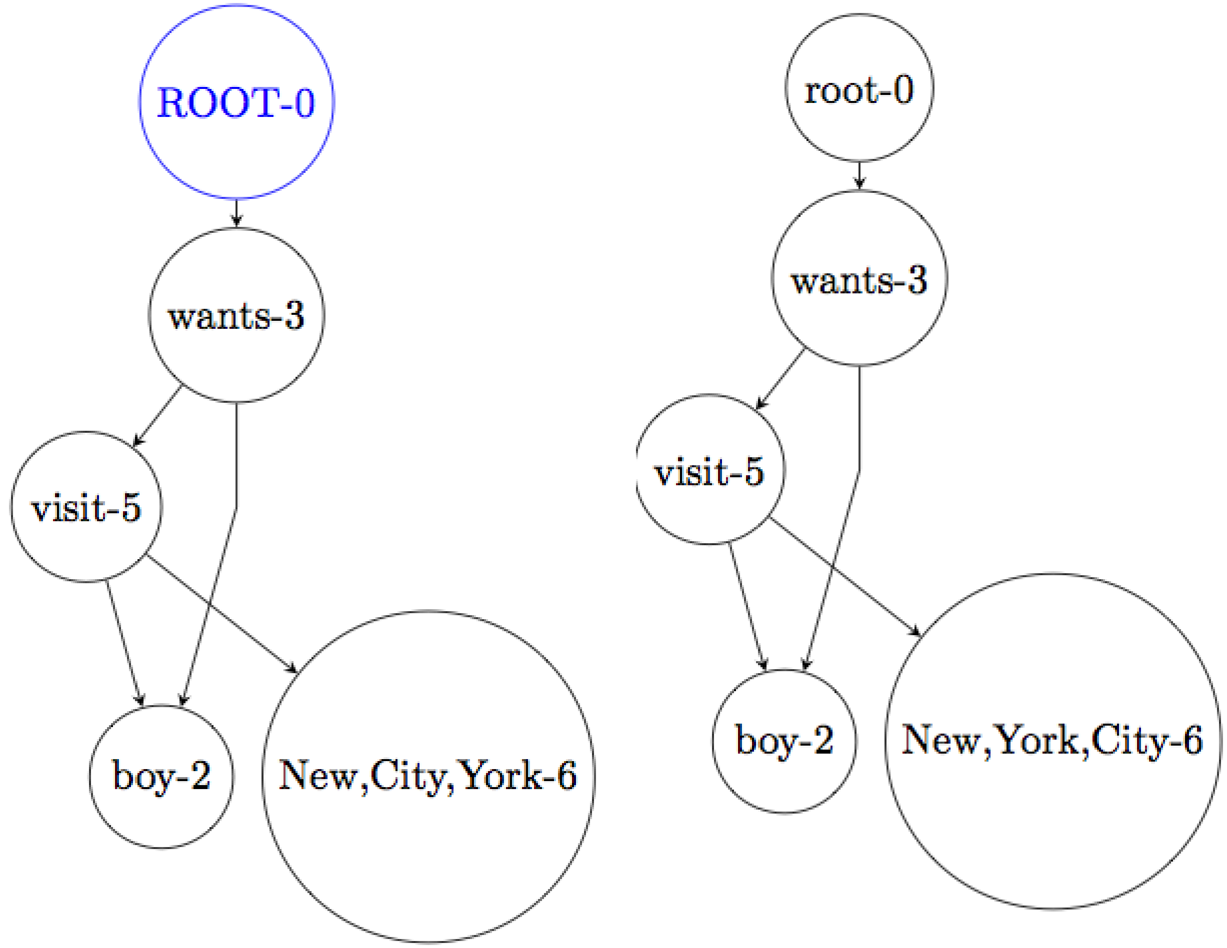
Example

σ:[Root-0|σ]
β:[wants-3|β]
Next



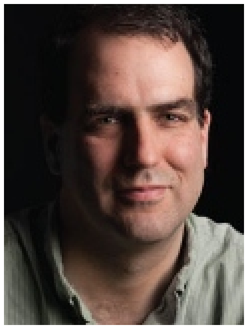


Example

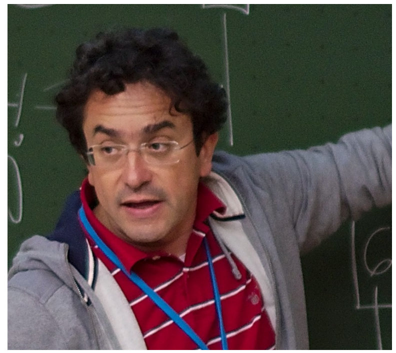


ROOT-0
wants-3
visit-5
boy-2
New,City,York-6
root-0
wants-3
visit-5
boy-2
New,York,City-6

Graph Learning for AMR (GLAMR)



Dan Gildea
(Rochester)



Giorgio Satta
(Padova)

David Chiang
(USC/ISI)

Frank Drewes
(Umea)

- Xiaochang Peng (Rochester)
- Naomi Saphra (JHU)

Graph Learning for AMR (GLAMR)

Syntax-Based MT:

string → tree → string synchronous context-free grammar

Semantics-Based MT:

string → graph → string ?

Hyperedge Replacement Grammars (HRG)

- Generalize Context-Free Grammars to generate graphs
  - Terminal and Nonterminal Hyperedges (Drewes et al., 1999)
- Parsing graph with HRG:
  - O(((3^d)n)^(k+1)) (Chiang et al., ACL 2013)
    - n: size of graph
    - d: degree of graph
    - k: treewidth of grammar
- Our goals this month:
  - Restricted formalism for NLP that is polynomial-time parsable
  - MCMC grammar learning

Dependency Tree to Graph Transition System

A transition system is a quadruples S = (C, T, Cs,Ct), 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: Ci→ Cj
3. Cs is an initialization function, mapping a sentence and its dependency tree to an initial parsing state
4. Ct is a set of terminal parsing state