Natural Language Processing

Lecture 16: Semantic Parsing II -
Abstract Meaning Representation (AMR)

11/20/2018

COMS W4705
Daniel Bauer

Logical Forms

- Logical form satisfies many goals for meaning representations (unambiguous, canonical form, supports inference, expressiveness)

- But difficult to annotate on a large scale.

Event Semantics

- Typically events and relations are expressed as predicates in first-order logic.

  $\exists x \; \exists y \; Computer(x) \wedge School(y) \wedge donate(Apple,x,y)$

- Problem: predicate arguments are not semantic roles.
  Model-theoretic semantics is problematic:

  donate(Apple,x,y) should imply *donate(Apple,x)*

- One approach: Event semantics (neo-Davisonian semantics).
  Events are treated like entities.

  $$\exists \mathbf{e} \; \exists x \; \exists y \; \boldsymbol{Donate(e)} \wedge Computer(x) \wedge School(y) \wedge$$
  $$Donor(\mathbf{e},Apple) \wedge Theme(\mathbf{e},x) \wedge Receipient(\mathbf{e},y)$$

- How is this related to frames and semantic roles?

Abstract Meaning Representation (AMR)

(Banarescu et al., 2013)

- Uses a single, simple data structure (feature structures / directed graphs) to represent many aspects of meaning.
- Focus on "who does what to whom" but leave out details (tense, quantifiers, etc.)
- This level of abstraction facilitates **consistent, large-scale human annotation.**
  Goal: build a giant "semantics bank" (comparable to treebanks for syntax).

(amr.isi.edu)

AMR Example

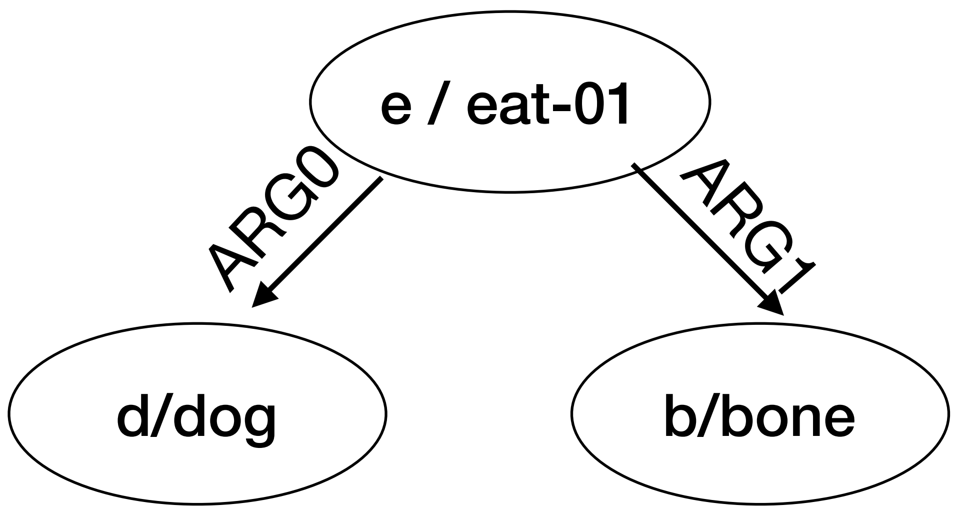
The dog is eating a bone.

```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```

AMR Example

The dog is eating a bone.

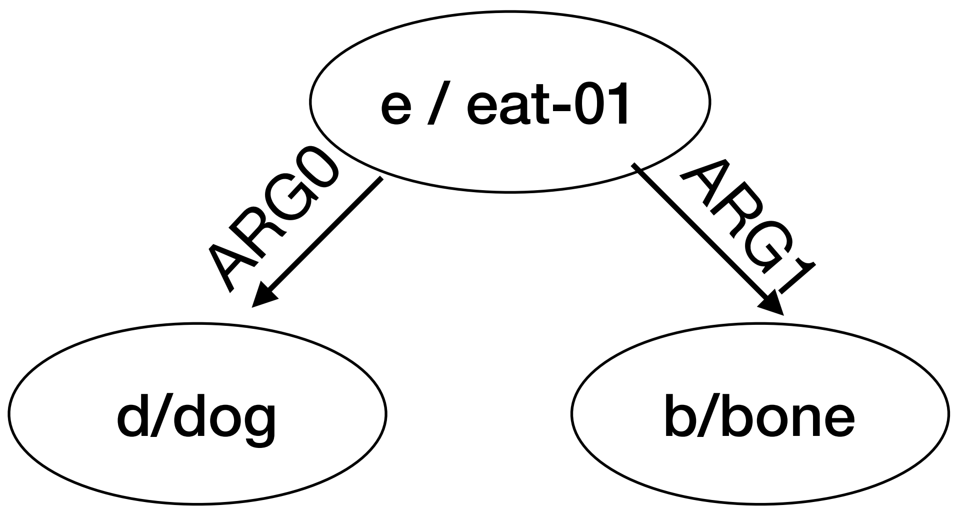
```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```



AMR Example

The dog is eating a bone.

```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```



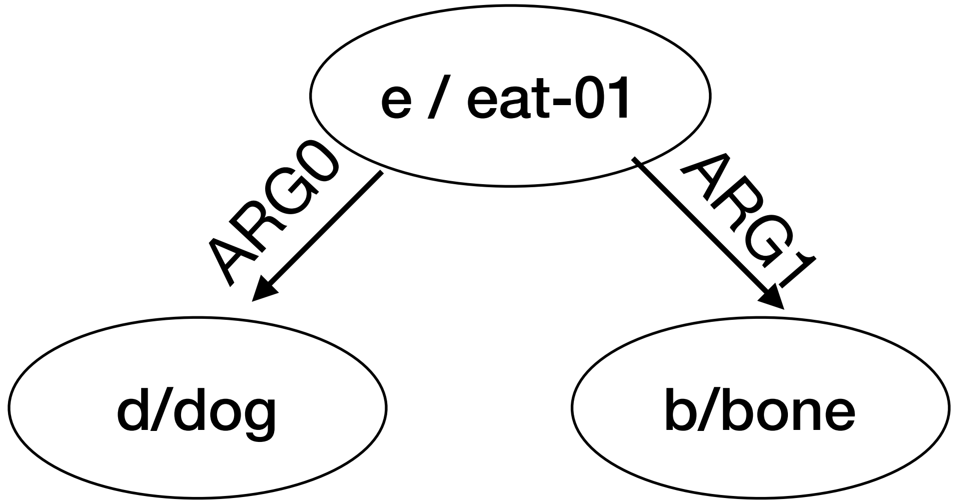
- Edges are labeled with **relations (including semantic roles)**
- Each node has a **variable.**
- Nodes are labeled with **concepts.**

- PropBank framesets used wherever possible.

AMR Example

The dog is eating a bone.

```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```



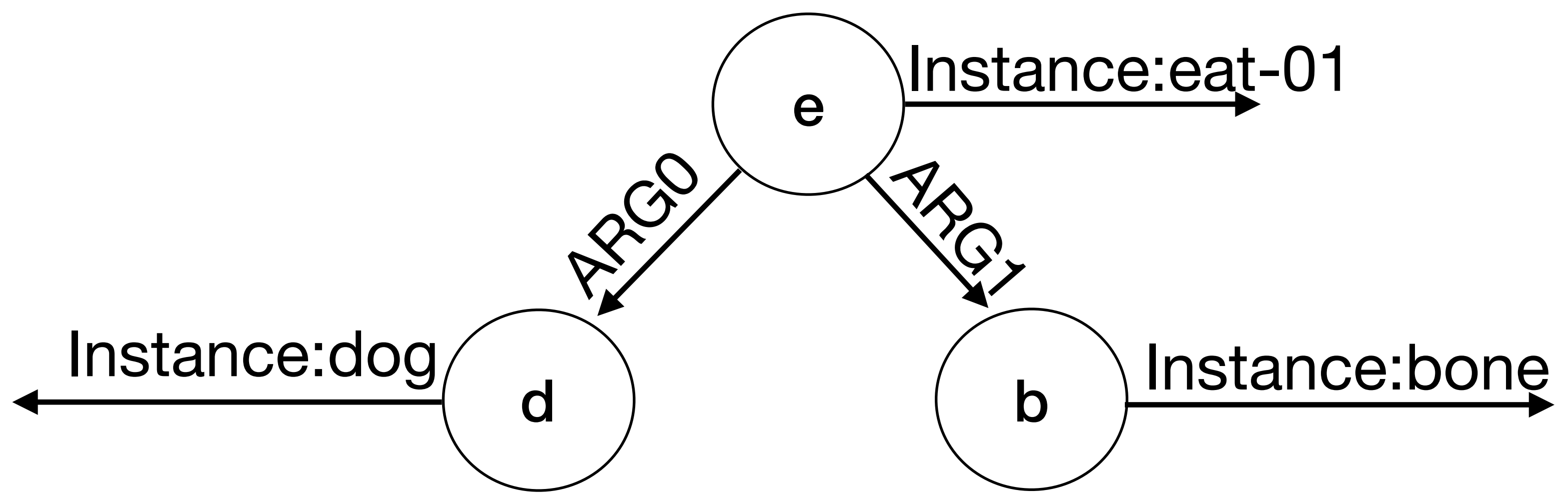
- Edges are labeled with **relations (including semantic roles)**
- Each node has a **variable.**
- Nodes are labeled with **concepts.**

- PropBank framesets used wherever possible.

AMR Example

The dog is eating a bone.

```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```



- Edges are labeled with **relations (including semantic roles)**
- Each node has a **variable.**
- Nodes are labeled with **concepts.**
    - Concepts can also be represented as edges.
- PropBank framesets used wherever possible.

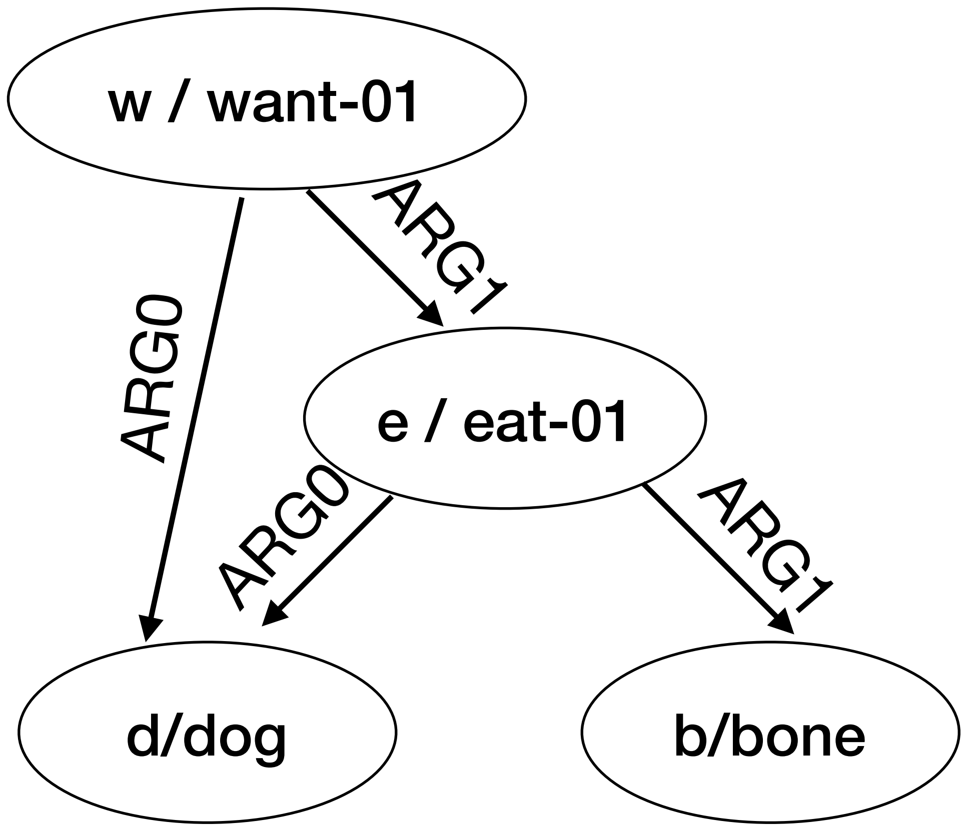
Reentrancy

The dog wants to eat a bone.

Reentrancy

The dog wants to eat a bone.

```
(w / want-01
    :ARG0 (d / dog)
    :ARG1 (e / eat-01
        :ARG0 d
        :ARG1 (b / bone))
```

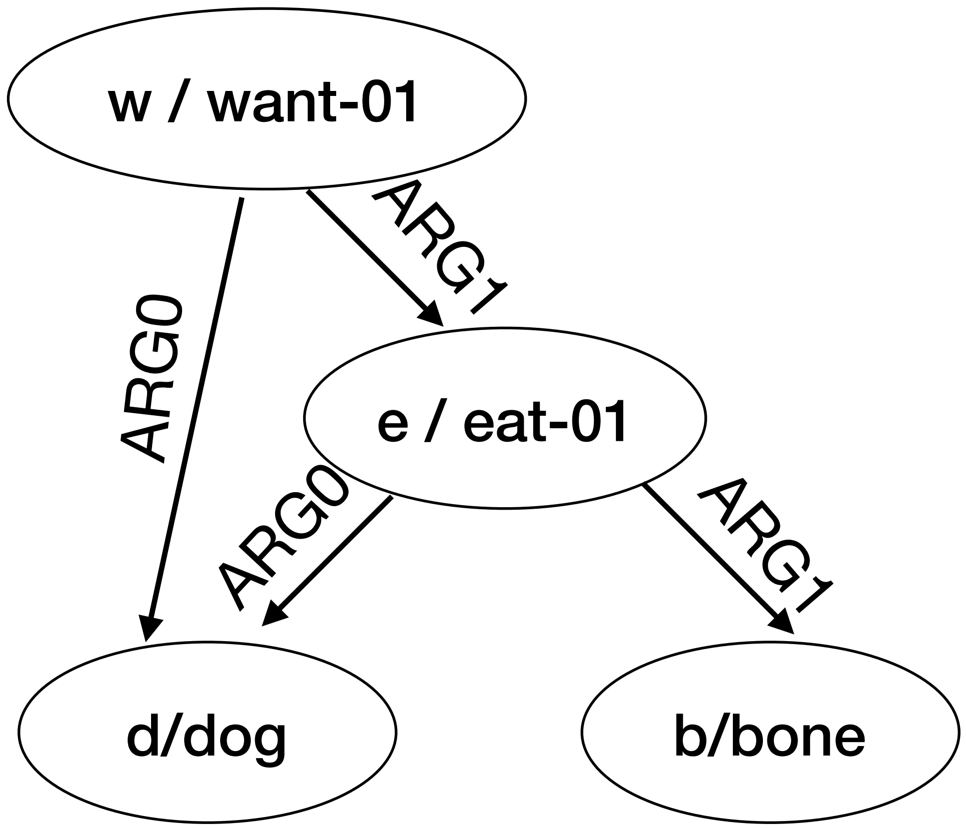


- Why the graph representation? Entities can play multiple roles.
- Two incoming edges in the graph, re-used variable in string notation.

AMR and Event Logic

The dog wants to eat a bone.

```
(w / want-01
    :ARG0 (d / dog)
    :ARG1 (e / eat-01
        :ARG0 d
        :ARG1 (b / bone)))
```

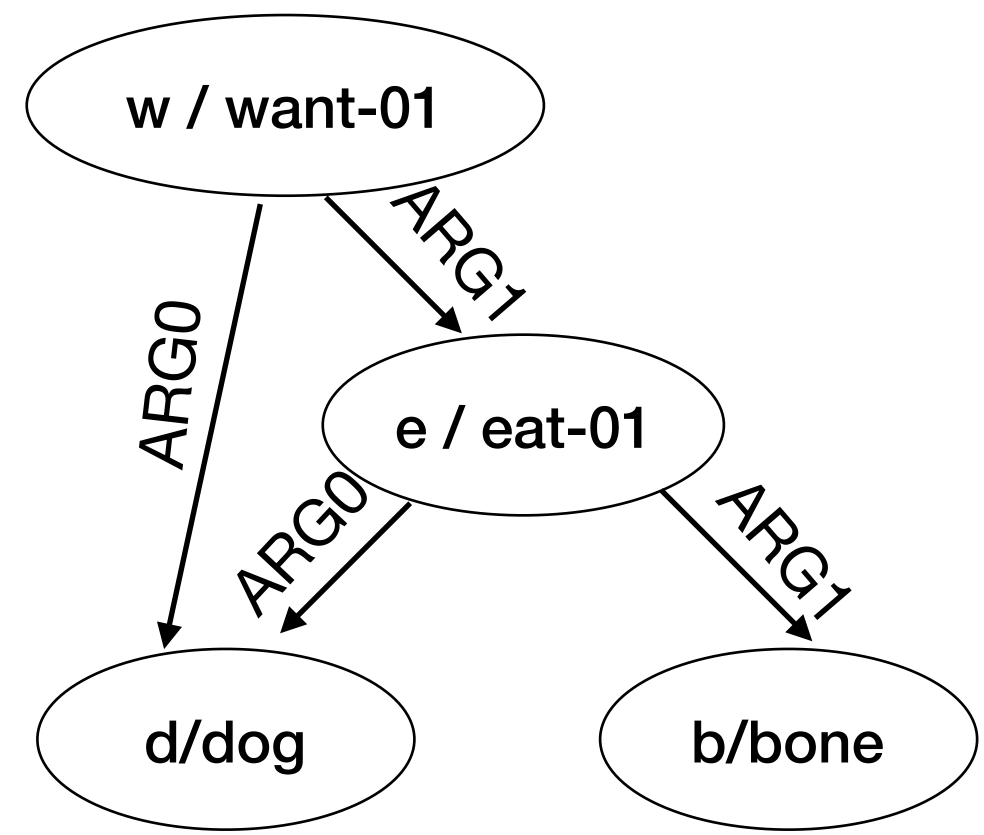


- AMR is related to event logic:
  - All concepts are existentially quantified.
  - Relations and concept labels are predicates.

AMR and Event Logic

The dog wants to eat a bone.

```
(w / want-01
    :ARG0 (d / dog)
    :ARG1 (e / eat-01
        :ARG0 d
        :ARG1 (b / bone))
```



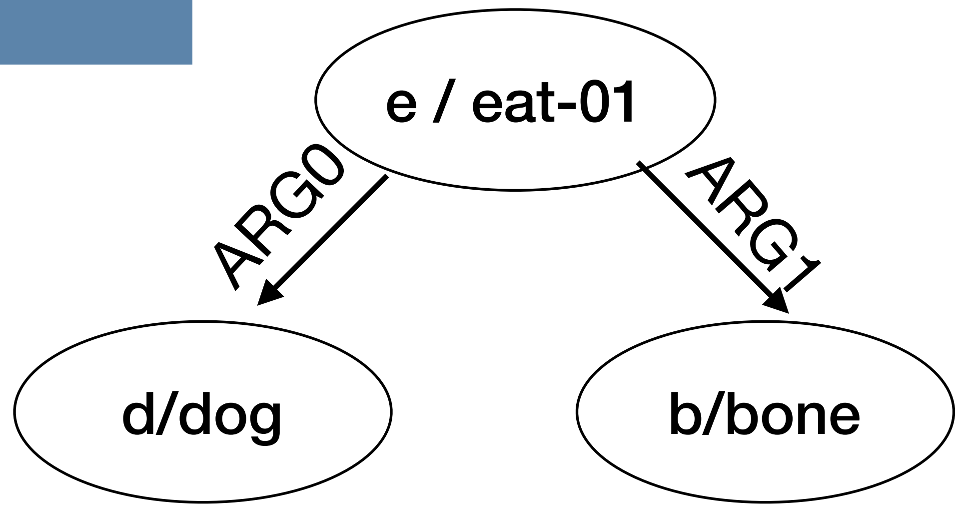
- AMR is related to event logic:
  - All concepts are existentially quantified.
  - Relations and concept labels are predicates.

$$\exists w\ \exists d\ \exists e\ \exists b\ \mathit{Want}(w) \wedge \mathit{Dog}(e) \wedge \mathit{Eat}(e) \wedge \mathit{Bone}(b) \wedge \mathit{ARG0}(w,d) \wedge \mathit{ARG1}(w,e) \wedge \mathit{ARG0}(e,d) \wedge \mathit{ARG1}(e,b)$$

Canonical Representaiton

The dog is eating a bone.
The bone was eaten by the dog.
The dog's eating of the bone.
...

```
(e / eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
```



- Many different sentences can have the same AMR representation.
- Nouns can describe events too.

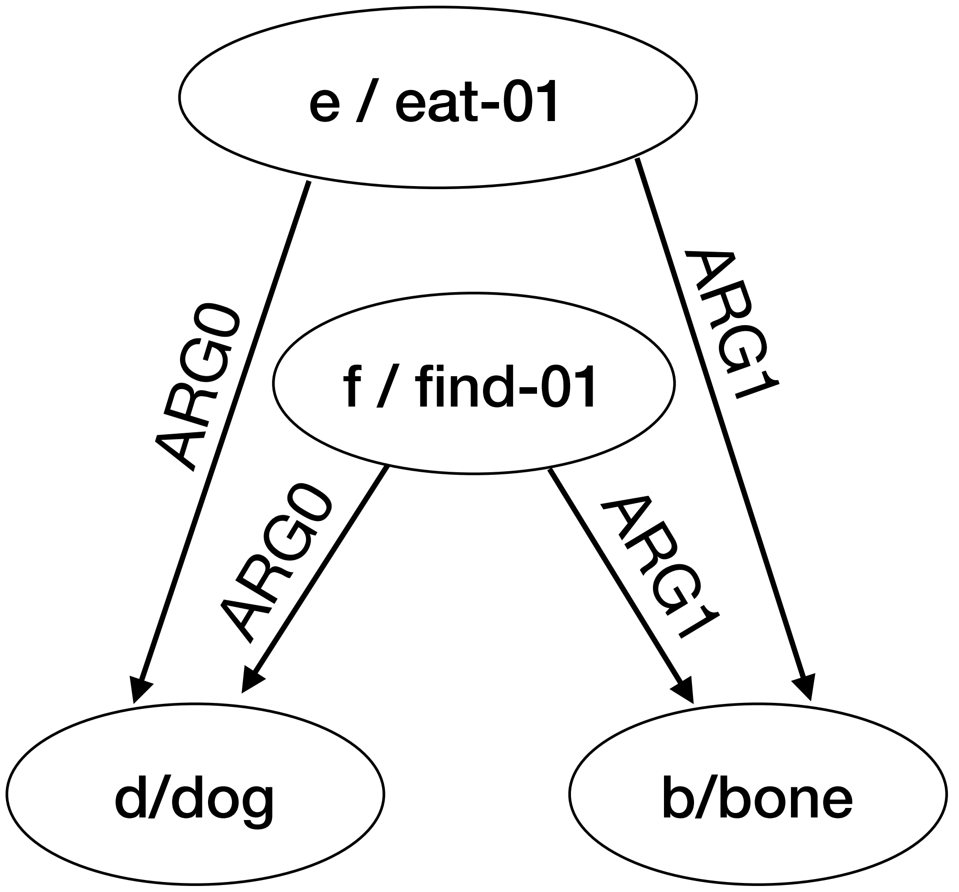
Inverse relations

The dog ate a bone that he found.

Inverse relations

The dog ate a bone that he found.

```
(e/ eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone))
(f/ find-01
    :ARG0 d
    :ARG1 b)
```

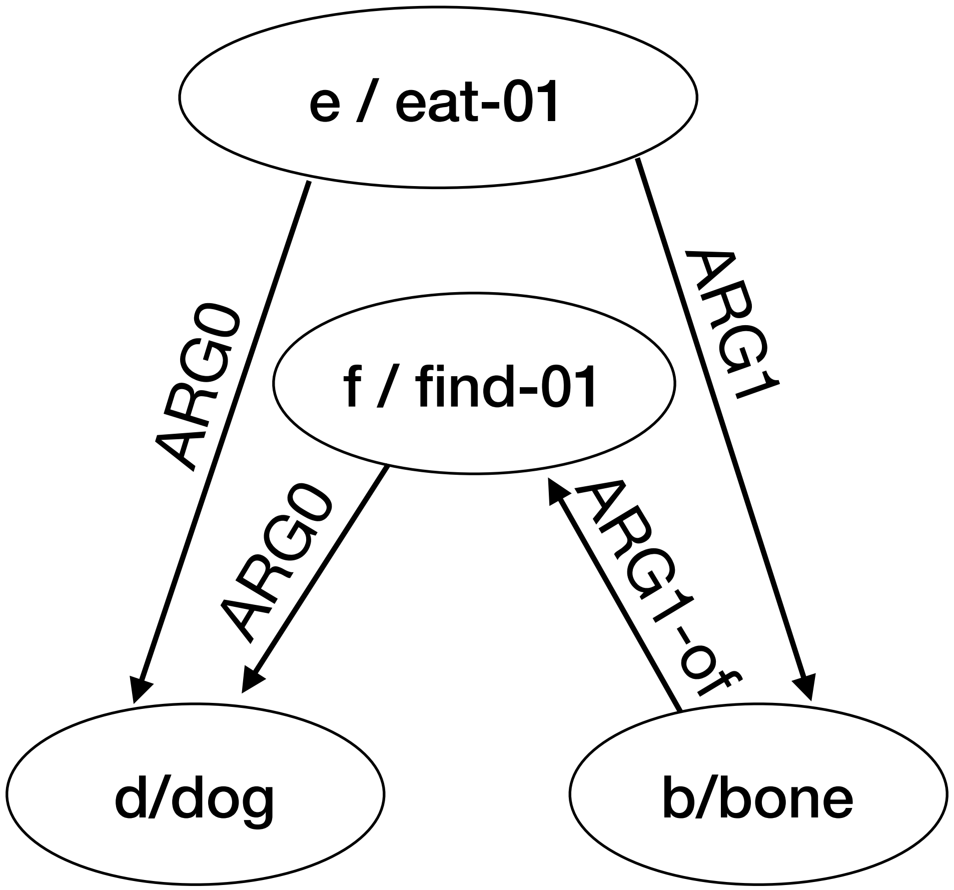


- AMR annotations are typically single-rooted (tree plus reentrancy)
- The single root is the "focus" of the sentence.

Inverse relations

The dog ate a bone that he found.

```
(e/ eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone
            :ARG1-of (f / find
                        :ARG0 d))
```

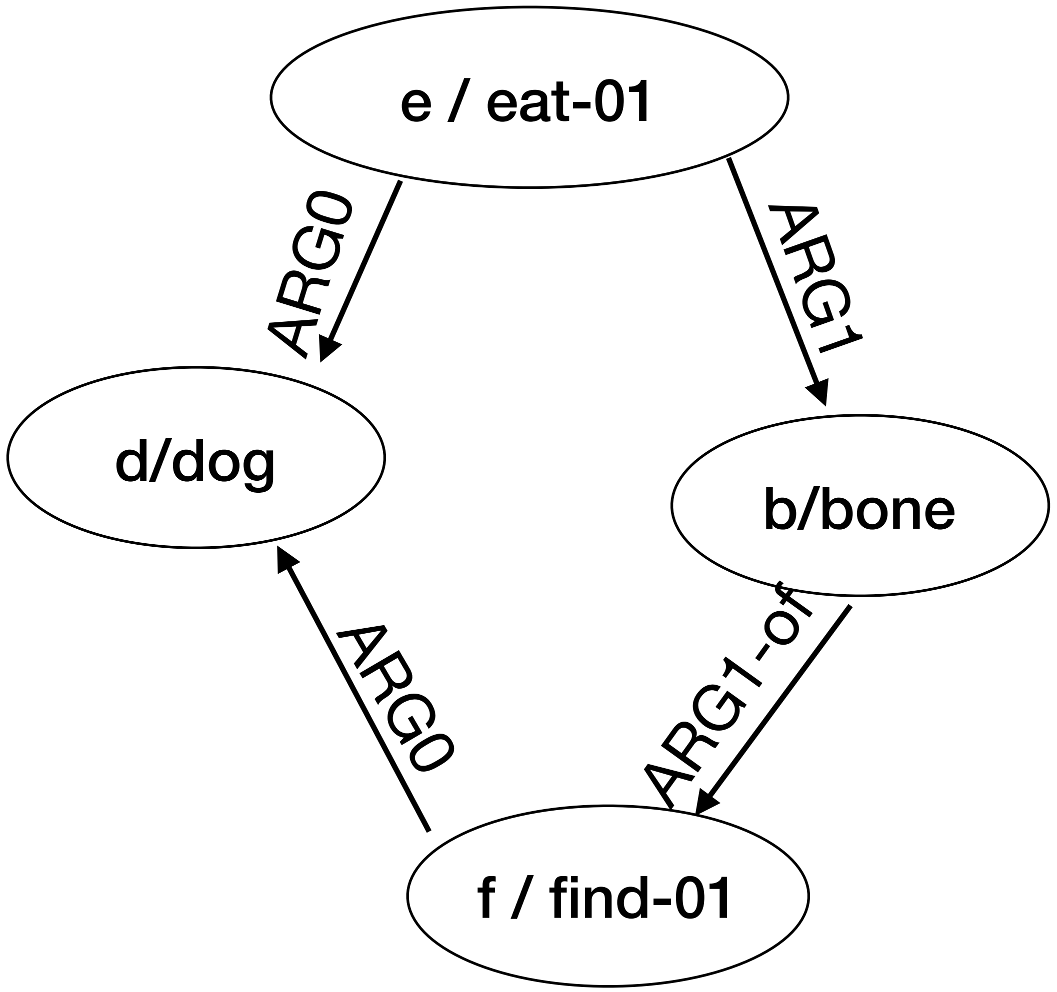


- AMR annotations are typically single-rooted (tree plus reentrancy)
- The single root is the "focus" of the sentence.

Inverse relations

The dog ate a bone that he found.

```
(e/ eat-01
    :ARG0 (d / dog)
    :ARG1 (b / bone
            :ARG1-of (f / find
                        :ARG0 d))
```

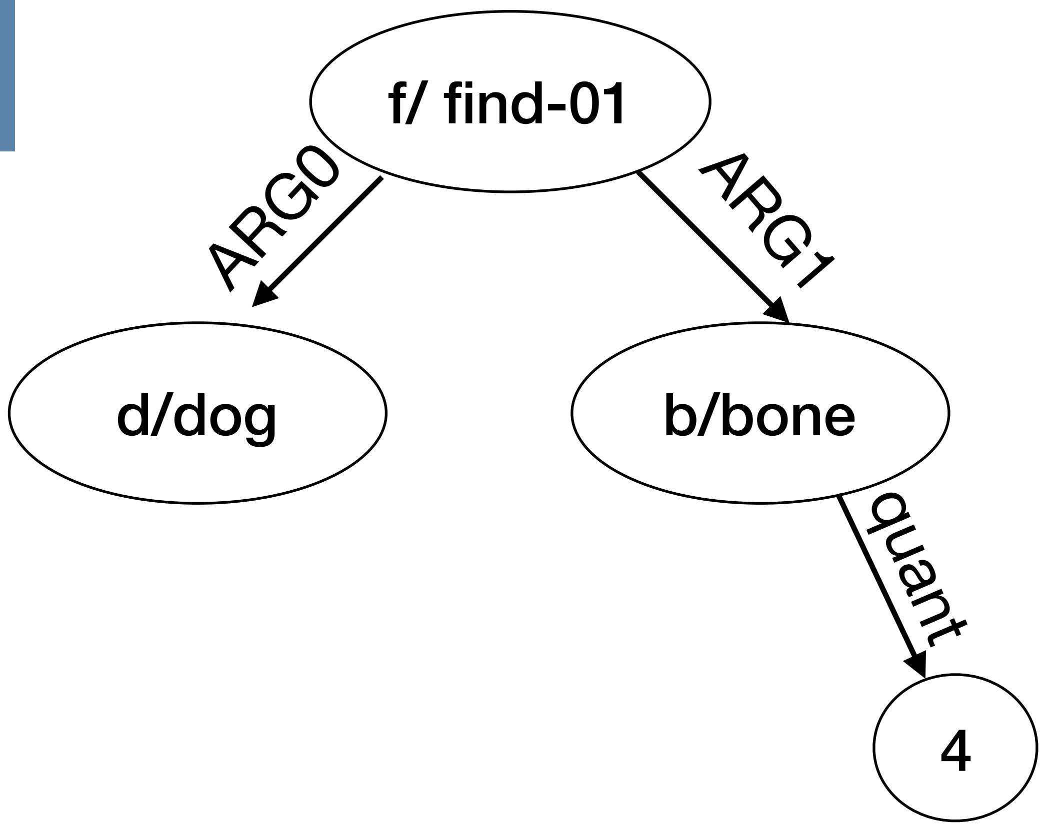


- AMR annotations are typically single-rooted (tree plus reentrancy)
- The single root is the "focus" of the sentence.

Constants

*The dog found **four** bones.*

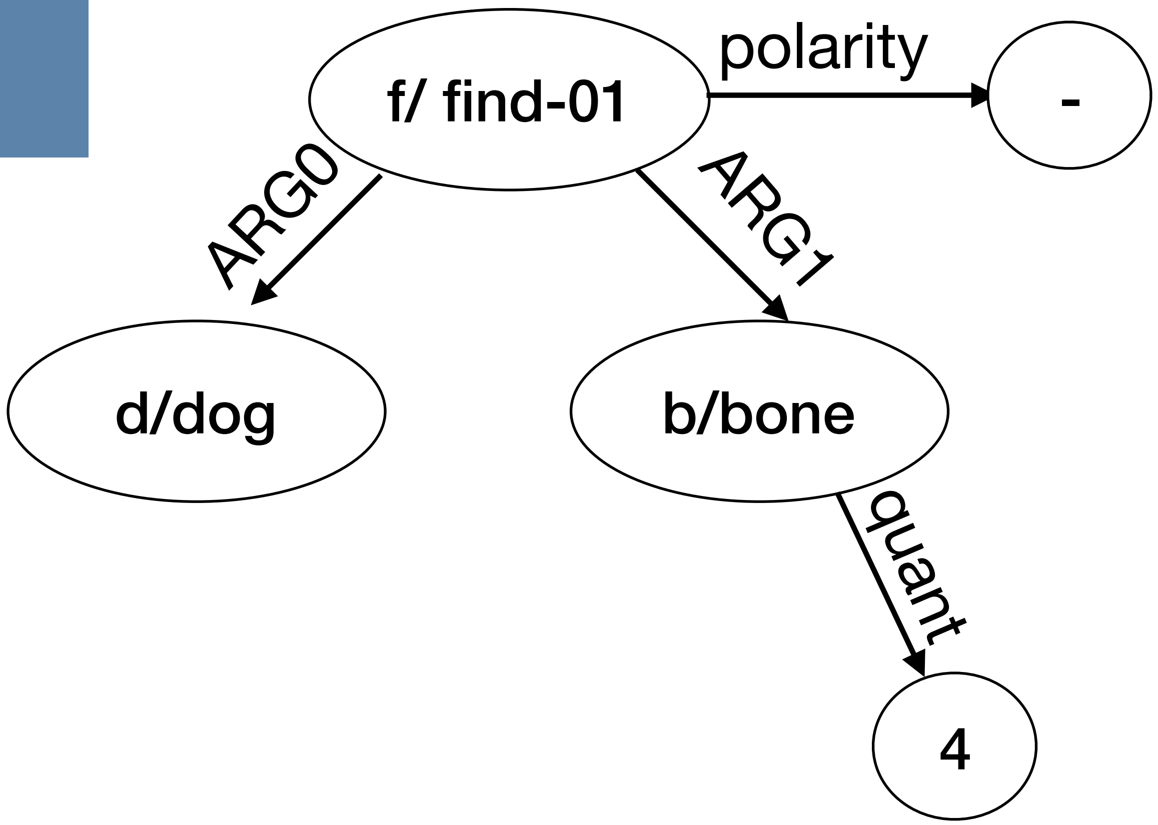
```
(f/ find-01
    :ARG0 (d / dog)
    :ARG1 (b / bone
            :quant 4))
```



- Constants are used to represent quantities (node gets no variable).
- Also used for negation.

Constants

*The dog did not find **four** bones.*




```
(f/ find-01
    :ARG0 (d / dog)
    :ARG1 (b / bone
           :quant 4)
    :polarity -)
```

- Constants are used to represent quantities (node gets no variable).
- Also used for negation.

Non-Core Roles

- AMR annotations use some built-in relations (not in PropBank)
  :time, :location, :manner, :part, :frequency
- :mod and :domain for attributes
- :op1, op2, ...for lists of arguments (for example in conjunctions).

```
(t/ truck
    :mod (m / monster))
```

a monster truck.

Non-Core Roles

- AMR annotations use some built-in relations (not in PropBank) :time, :location, :manner, :part, :frequency
- :mod and :domain for attributes
- :op1, op2, ...for lists of arguments (for example in conjunctions).

```
(t/ truck
   :mod (m / monster))
```

a monster truck.

```
(s/see-01
  (y / yummy
     :domain(f / food))
```

seeing that the food is yummy.

Non-Core Roles

- AMR annotations use some built-in relations (not in PropBank) :time, :location, :manner, :part, :frequency
- :mod and :domain for attributes
- :op1, op2, ...for lists of arguments (for example in conjunctions).

```
(t/ truck
   :mod (m / monster))
```

a monster truck.

```
(s/see-01
  (y / yummy
     :domain(f / food))
```

seeing that the food is yummy.

```
 (a / and
    :op1 (a / apple)
    :op2 (o / orange))
```

apples and oranges.

Names and Dates

```
(j / join-01
      :ARG0 (p / person :wiki -
            :name (p2 / name :op1 "Pierre" :op2 "Vinken")
            :age (t / temporal-quantity :quant 61
                  :unit (y / year)))
      :ARG1 (b / board
            :ARG1-of (h / have-org-role-91
                  :ARG0 p
                  :ARG2 (d2 / director
                        :mod (e / executive :polarity -))))
      :time (d / date-entity :month 11 :day 29))
```

AMR to English

```
(r / read-01
      :arg0 (j / judge)
      :arg1 (t / thing
             :arg1-of (p /propose-01))
```

AMR to English

```
(r / read-01
      :arg0 (j / judge)
      :arg1 (t / thing
               :arg1-of (p /propose-01))


(p / picture-01
  :ARG0 (i / it)
  :ARG1 (b2 / boa
           :mod (c / constrictor)
           :ARG0-of (d / digest-01
                        :ARG1 (e / elephant))))
```

English to AMR

- "*The girl wants the boy to like her*"
- "*The girl wants the boy to believe that she likes him*"

AMR Data

- The Little Prince
  (publicly available, http://amr.isi.edu/download.html):
  - English and Chinese
  - Biomedical Data
- "AMRBank", 14k sentence, PTB and other corpora
  (including online discussion forums)

Another AMR Example

Back downtown, the execs squeezed in a few meetings at the hotel before boarding the buses again.

Another AMR Example

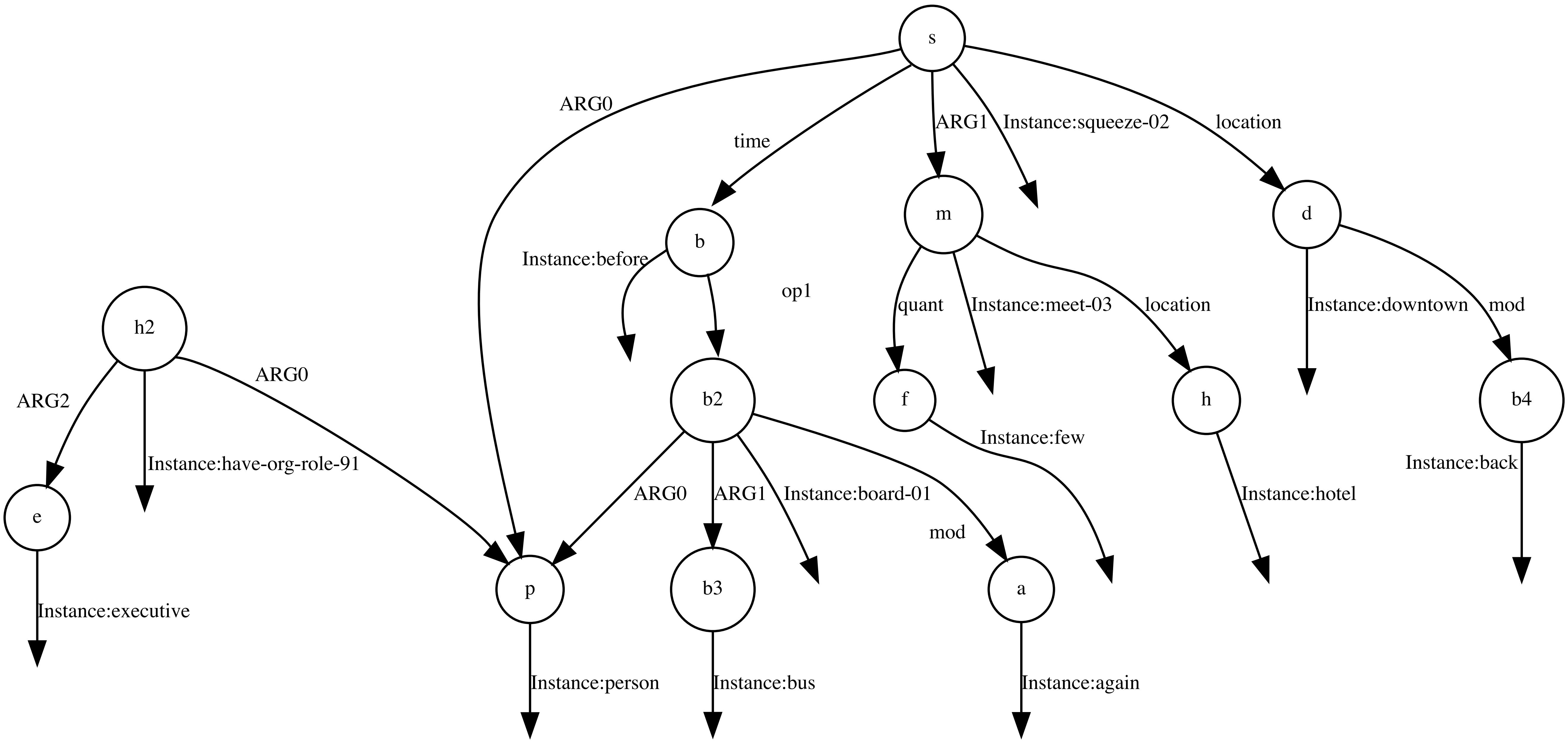
```
(s / squeeze-02
   :ARG0 (p / person
            :ARG0-of (h2 / have-org-role-91
                         :ARG2 (e / executive)))

   :ARG1 (m / meet-03
           :location (h / hotel)
           :quant (f / few))


   :location (d / downtown
                :mod (b4 / back))
   :time (b / before
           :op1 (b2 / board-01
                    :ARG0 p
                    :ARG1 (b3 / bus)
                    :mod (a / again))))
```

Back downtown, the execs squeezed in a few meetings at the hotel before boarding the buses again.

Another AMR Example



Back downtown, the execs squeezed in a few meetings at the hotel before boarding the buses again.

Applications of AMR

- Semantics-Based Machine Translation
  (Jones, Andreas, Bauer, Hermann & Knight, 2012)
- Summarization:
  - Abstractive Summarization
    (Liu, Flanigan, Thomson, Sadeh & Smith, 2015)
  - Text Compression (text-to-text generation)
    (Thadani, 2015)
- Predicting stock price movement from financial news
  (Xie, 2015)

AMR Parsing

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:
  - Concept identification + graph-based dependency parsing (JAMR, Flanigan et al. 2014)

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:
  - Concept identification + graph-based dependency parsing (JAMR, Flanigan et al. 2014)
  - Transduce dependency tree into AMR (Wang et al. 2015)

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:
  - Concept identification + graph-based dependency parsing (JAMR, Flanigan et al. 2014)
  - Transduce dependency tree into AMR (Wang et al. 2015)
  - CCG into logical form + coreference resolution (Artzi 2015)

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:
  - Concept identification + graph-based dependency parsing (JAMR, Flanigan et al. 2014)
  - Transduce dependency tree into AMR (Wang et al. 2015)
  - CCG into logical form + coreference resolution (Artzi 2015)
  - Syntax-based Machine Translation (Pust et al. 2015)

AMR Parsing

- English -> AMR, trained automatically on string/graph pairs (AMR data)
- Different approaches:
  - Concept identification + graph-based dependency parsing (JAMR, Flanigan et al. 2014)
  - Transduce dependency tree into AMR (Wang et al. 2015)
  - CCG into logical form + coreference resolution (Artzi 2015)
  - Syntax-based Machine Translation (Pust et al. 2015)
  - Hyperedge Replacement Grammars (Peng 2015, Bauer 2017)

JAMR

(Flanigan et al. 2014)

- Automatically align string spans and graph concepts to obtain a **concept dictionary.**
- For an unseen input sentence:
    - Identify the concepts in the sentence.
    - Identify the relations (edges) between the concepts using a graph-based aproach ("spanning graph").
        - Similar to graph-based dependency parsing.

JAMR - Alignments

(Flanigan et al. 2014)

- Uses a set of hand-crafter rules (patterns for named entities, dates, ...)
- Goal: Compute a concept dictionary.

(Schneider & Flanigan, AMR tutorial @ NAACL 2015)

JAMR - Alignments

(Flanigan et al. 2014)

- Uses a set of hand-crafter rules (patterns for named entities, dates, ...)
- Goal: Compute a concept dictionary.

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
    :ARG0 (o / organization
        :ARG0 (n / name
            :op1 “IAEA”))
    :ARG1 (t2 / thing
        :ARG1-of (p / propose-01
            :ARG0 (c / country
                :name (n2 / name
                    :op1 “North”
                    :op2 “Korea”))))
    :time (d / date-entity
        :month 11))
```

(Schneider & Flanigan, AMR tutorial @ NAACL 2015)

JAMR - Concept ID

The boy wants to visit New York City

Figure 2: A concept labeling for the sentence "The boy wants to visit New York City."

JAMR - Concept ID

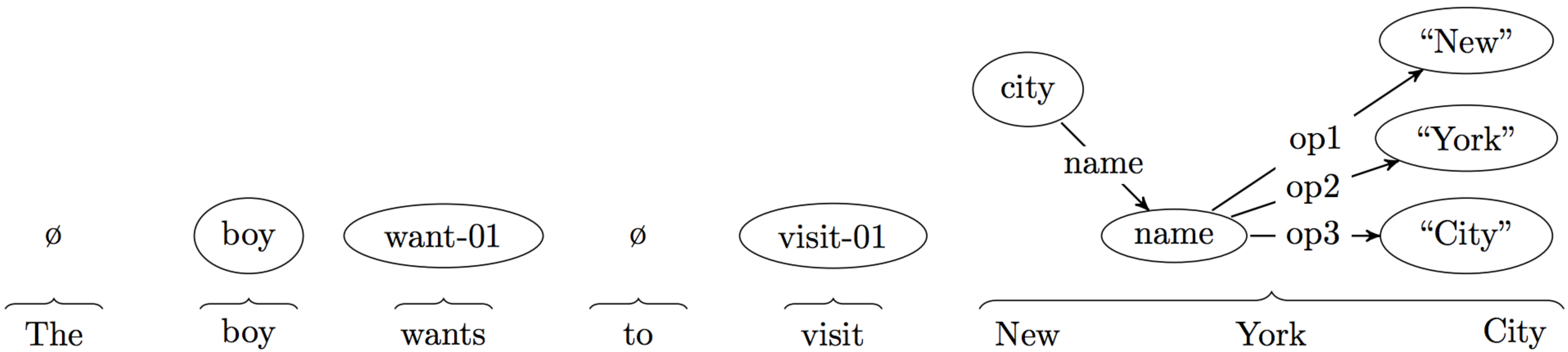
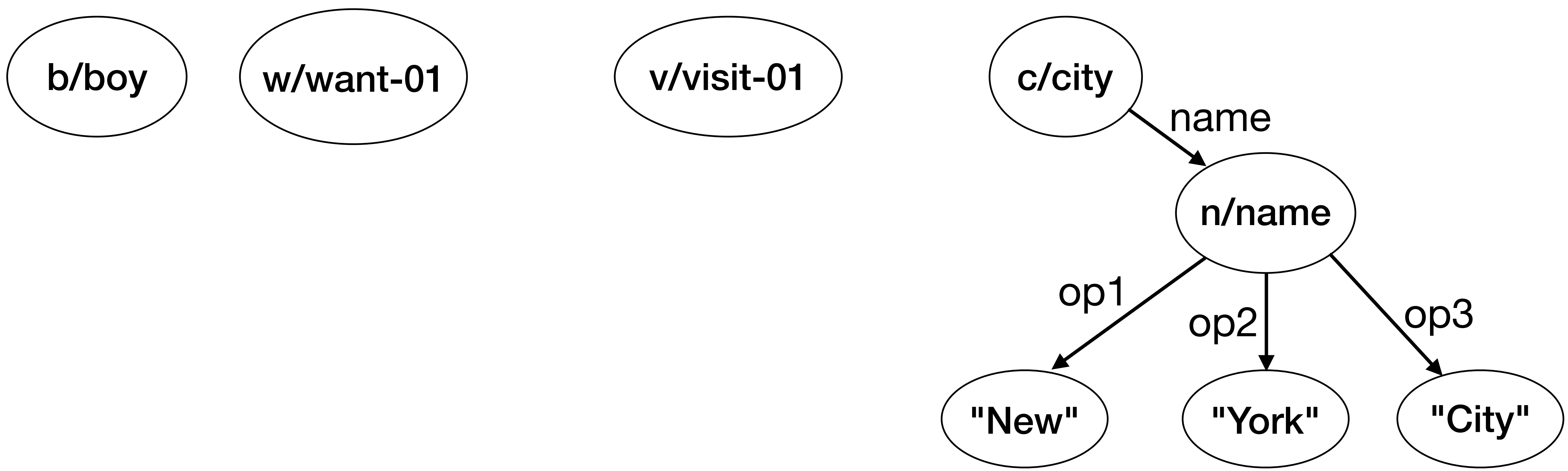


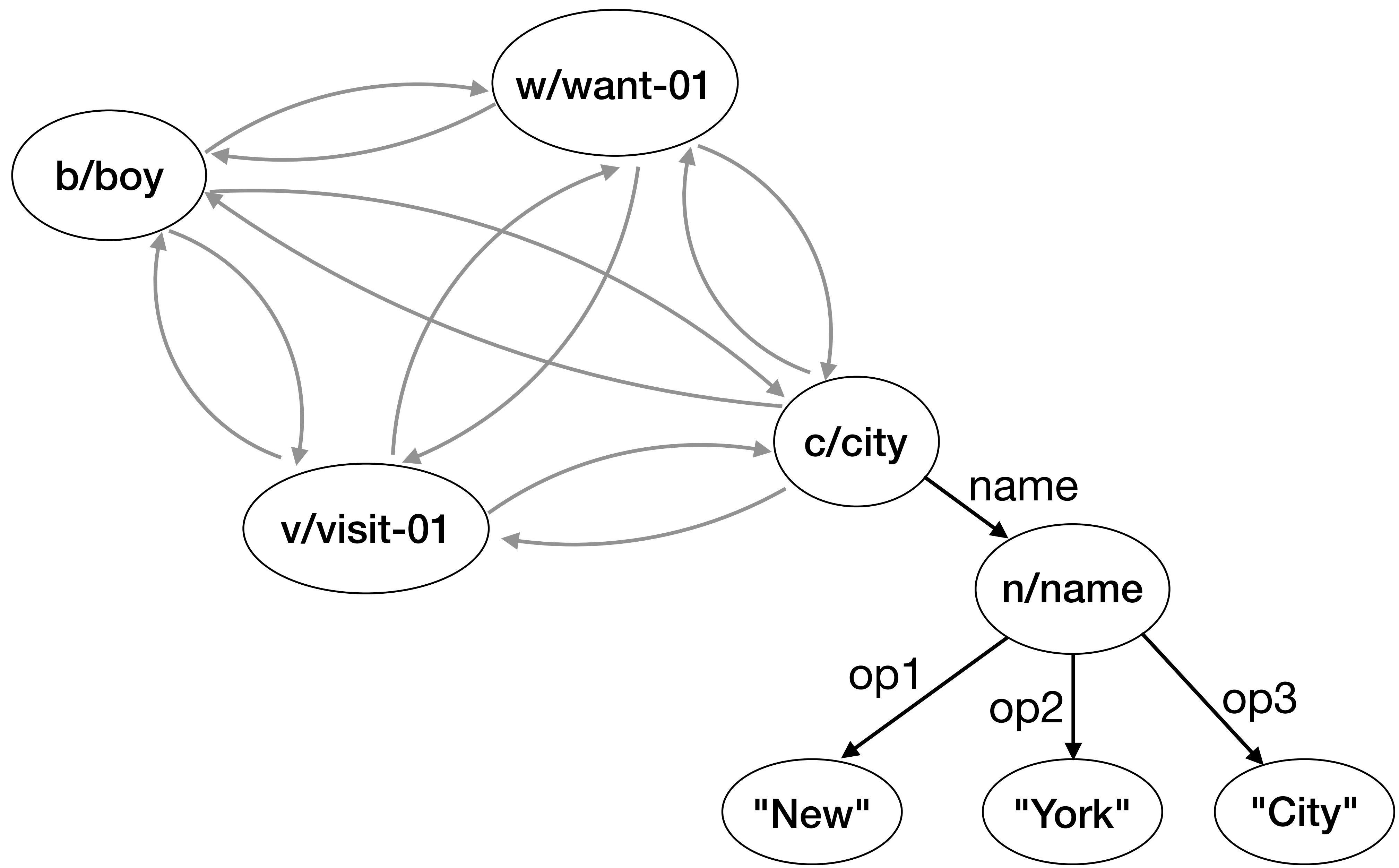
Figure 2: A concept labeling for the sentence "The boy wants to visit New York City."

- Need to compute best span-to-concept assignment.
- But need to consider all different spans.
- Dynamic programing algorithm to solve this.

JAMR - Relation ID

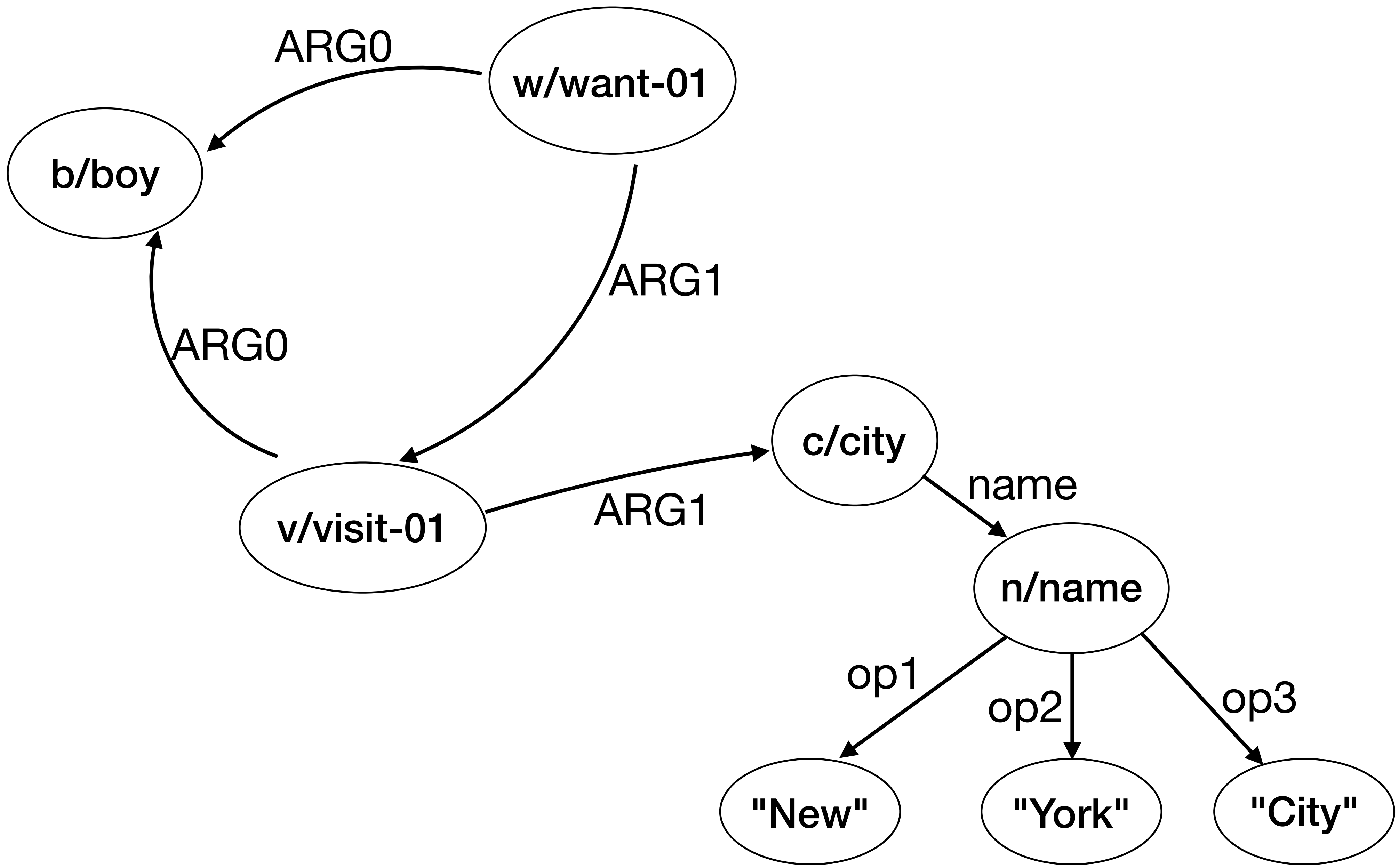


JAMR - Relation ID



Start with completely connected graph (one edge for each relation).

JAMR - Relation ID



Then compute the "Maximum Spanning Connected Subgraph" (MSCC)

AMR Parsing with Synchronous Hyperedge Replacement Grammar

(Bauer 2017)

AMR Parsing with Synchronous Hyperedge Replacement Grammar

(Bauer 2017)

- String CFGs assemble strings, Hyperedge Replacement Grammars (HRG) assemble graphs.

AMR Parsing with Synchronous Hyperedge Replacement Grammar

(Bauer 2017)

- String CFGs assemble strings, Hyperedge Replacement Grammars (HRG) assemble graphs.

- Use grammar rules with two matching right-hand sides (string, graph).

  - Parse the string, then follow the derivation and assemble a graph.

AMR Parsing with Synchronous Hyperedge Replacement Grammar

(Bauer 2017)

- String CFGs assemble strings, Hyperedge Replacement Grammars (HRG) assemble graphs.

- Use grammar rules with two matching right-hand sides (string, graph).

    - Parse the string, then follow the derivation and assemble a graph.

- The actual formalism is closer to TAG (lexicalized, allows an adjunction-like operation).

AMR Parsing with Synchronous Hyperedge Replacement Grammar

(Bauer 2017)

- String CFGs assemble strings, Hyperedge Replacement Grammars (HRG) assemble graphs.

- Use grammar rules with two matching right-hand sides (string, graph).

    - Parse the string, then follow the derivation and assemble a graph.

- The actual formalism is closer to TAG (lexicalized, allows an adjunction-like operation).

- Rules obtained automatically using alignments.

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree

Derived String

S

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

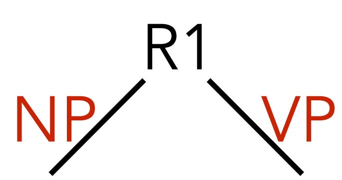
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

S

Recall: String CFG

R1: S →NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

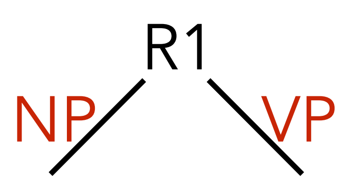
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

NP VP

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

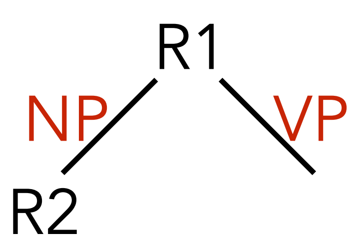
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

I VP

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

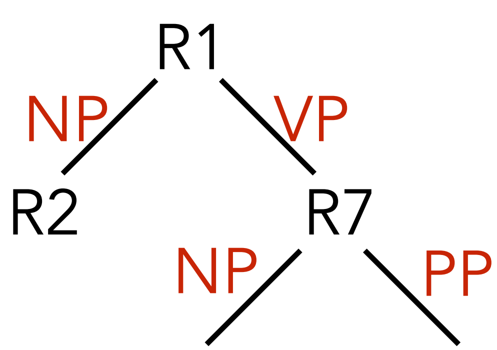
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

I shot NP PP

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

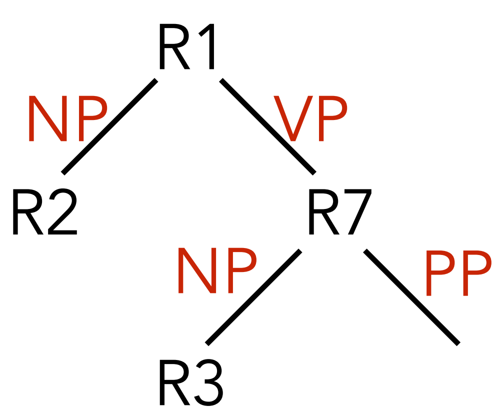
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

I shot an elephant PP

Recall: String CFG

R1: S →NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

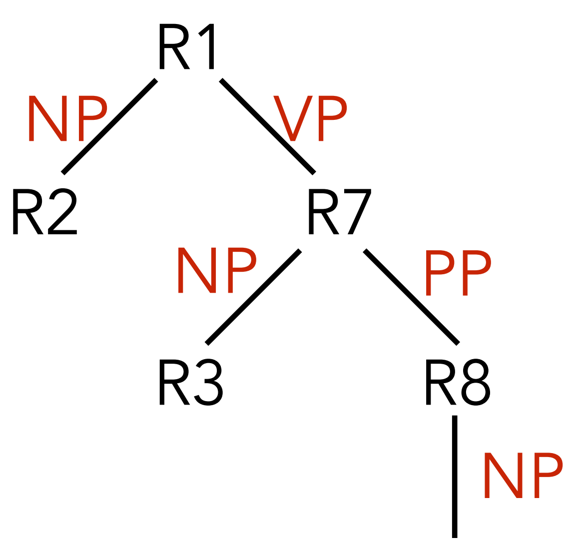
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

Derivation Tree



Derived String

I shot an elephant in NP

Recall: String CFG

R1: S → NP VP

R2: NP → *I*

R3: NP → *an elephant*

R4: NP → *my pajamas*

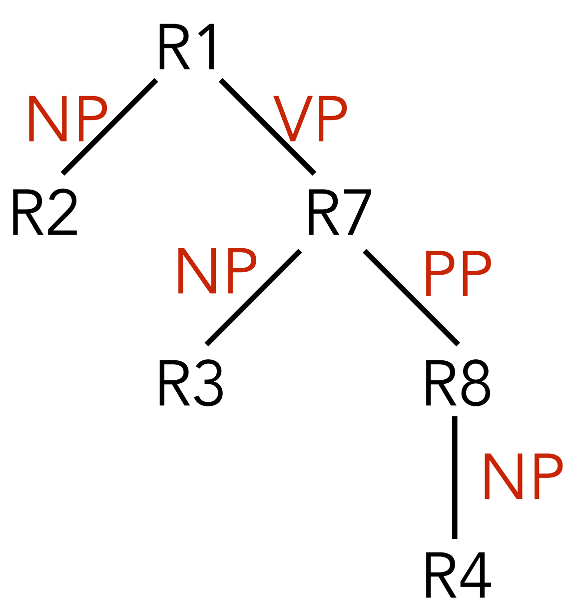
R5: VP → *shot* NP

R6: NP → NP PP

R7: VP → *shot* NP PP

R8: PP → *in* NP

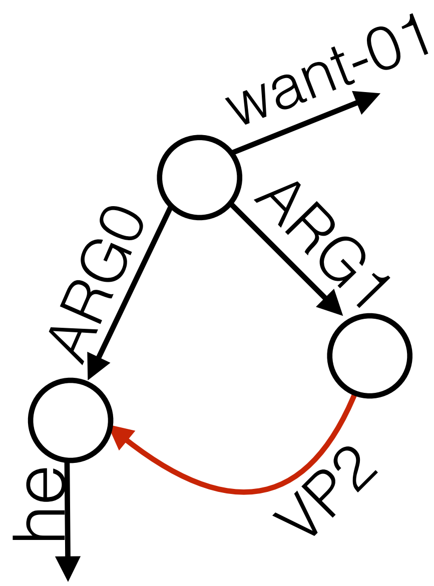
Derivation Tree

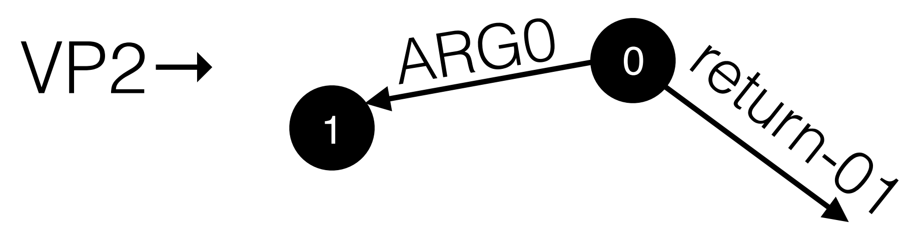


Derived String

I shot an elephant in my pajamas

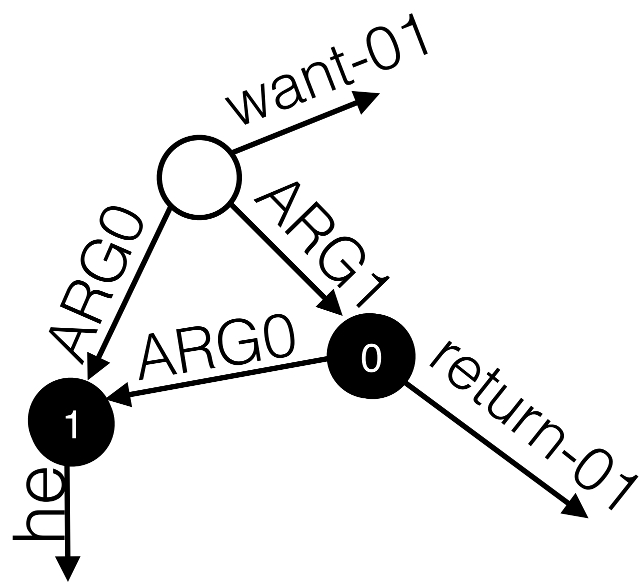
Hyperedge Replacement Grammar (HRG)



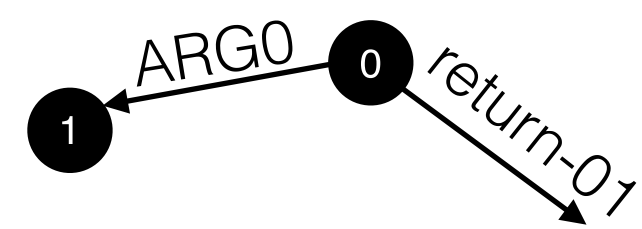


(Drewes et al. 1997)

Hyperedge Replacement Grammar (HRG)



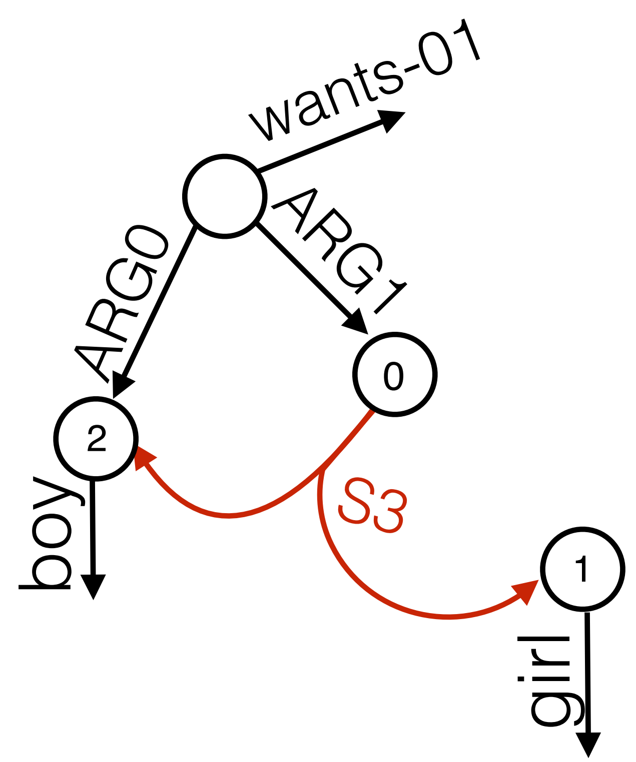
VP2→

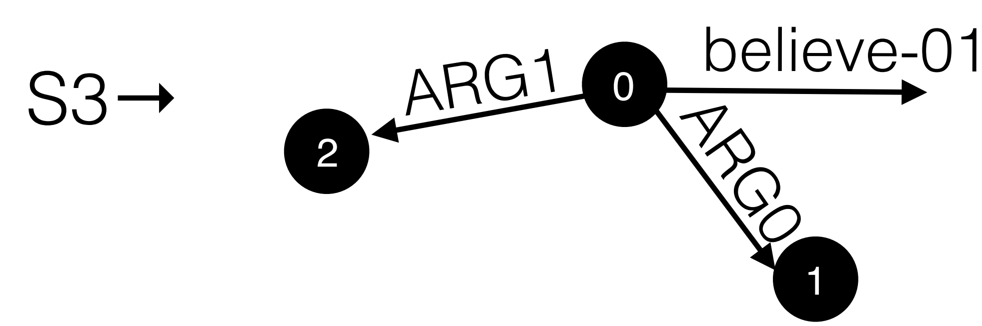


(Drewes et al. 1997)

Hyperedge Replacement Grammar (HRG)

- Hyperedges can have arbitrarily many tentacles. Their endpoints are ordered.

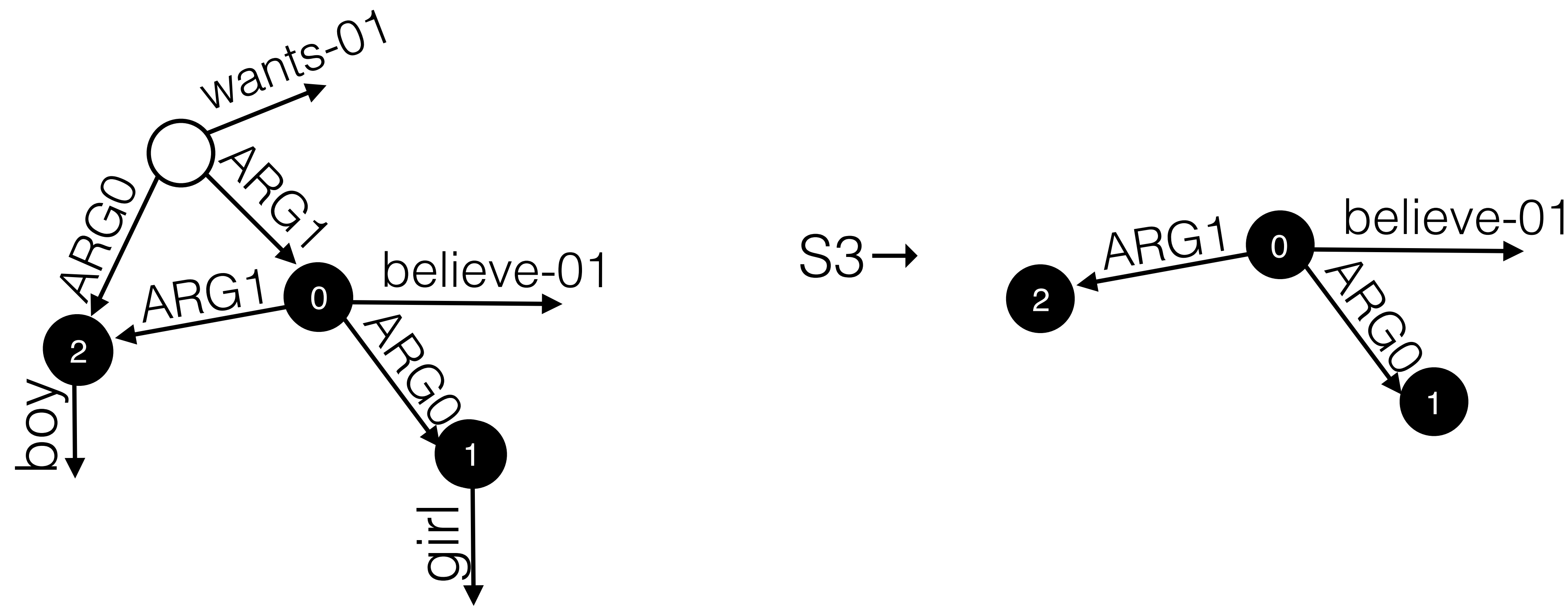




- Number of tentacles: **type** of the hyperedge.
- Terminal / Nonterminal alphabet is also typed.

Hyperedge Replacement Grammar (HRG)

- Hyperedges can have arbitrarily many tentacles. Their endpoints are ordered.



- Number of tentacles: **type** of the hyperedge.
- Terminal / Nonterminal alphabet is also typed.

Hyperedge Replacement Grammar (HRG)

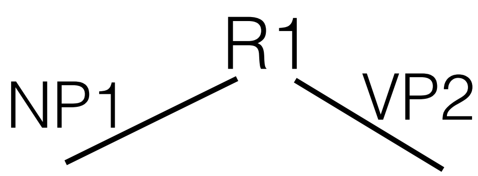
Derivation Tree

Derived Graph

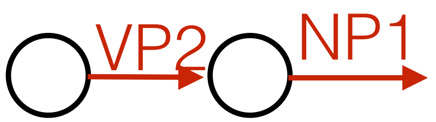
S0

Hyperedge Replacement Grammar (HRG)

Derivation Tree

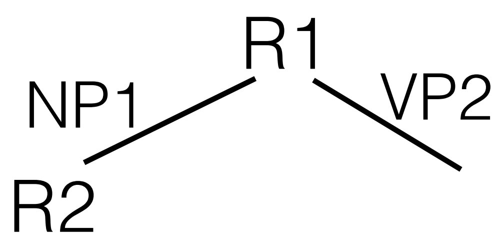


Derived Graph

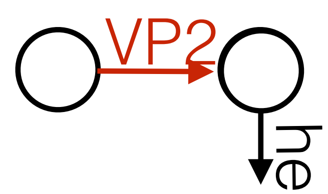


Hyperedge Replacement Grammar (HRG)

Derivation Tree

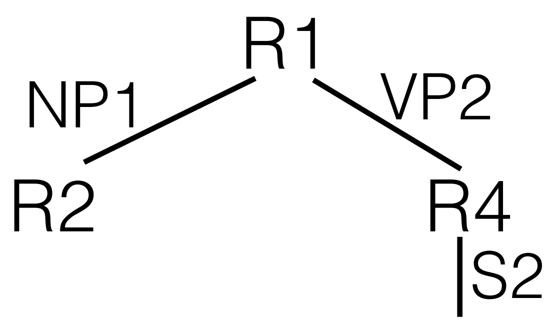


Derived Graph

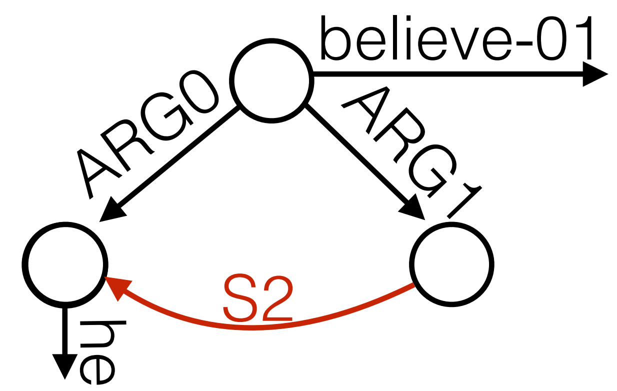


Hyperedge Replacement Grammar (HRG)

Derivation Tree

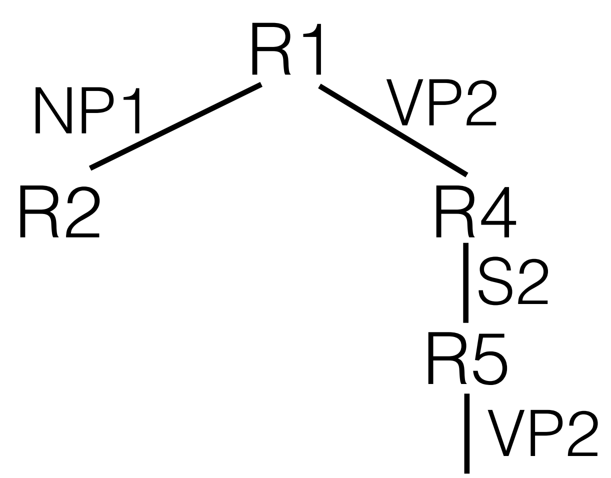


Derived Graph

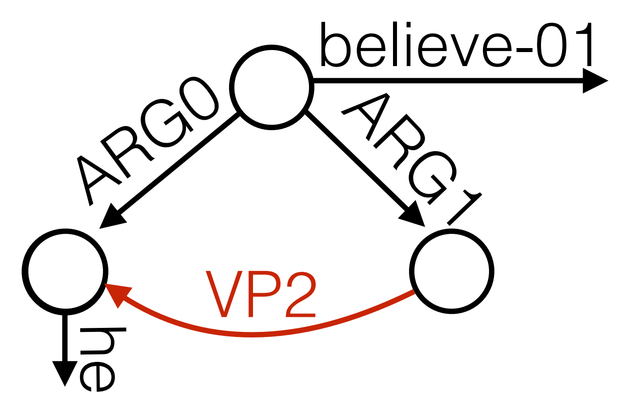


Hyperedge Replacement Grammar (HRG)

Derivation Tree

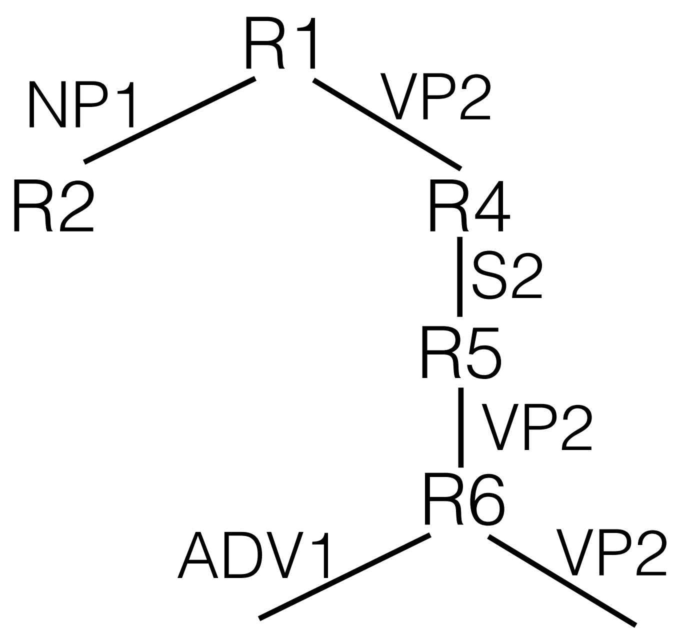


Derived Graph

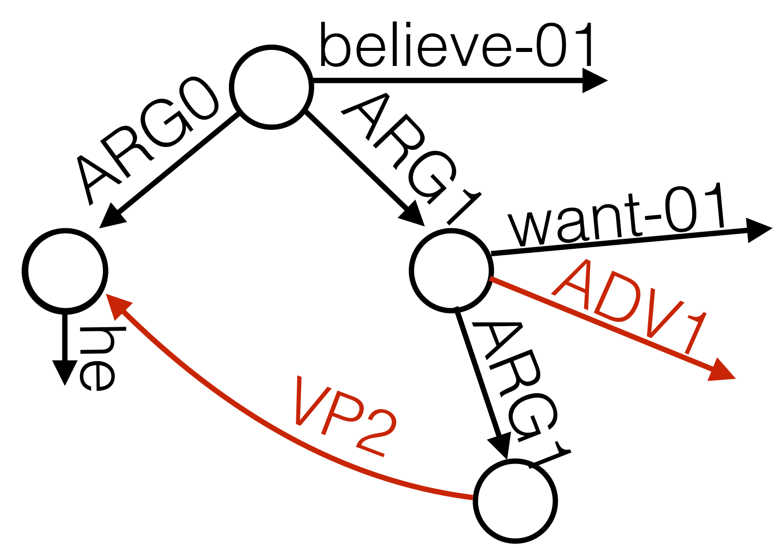


Hyperedge Replacement Grammar (HRG)

Derivation Tree

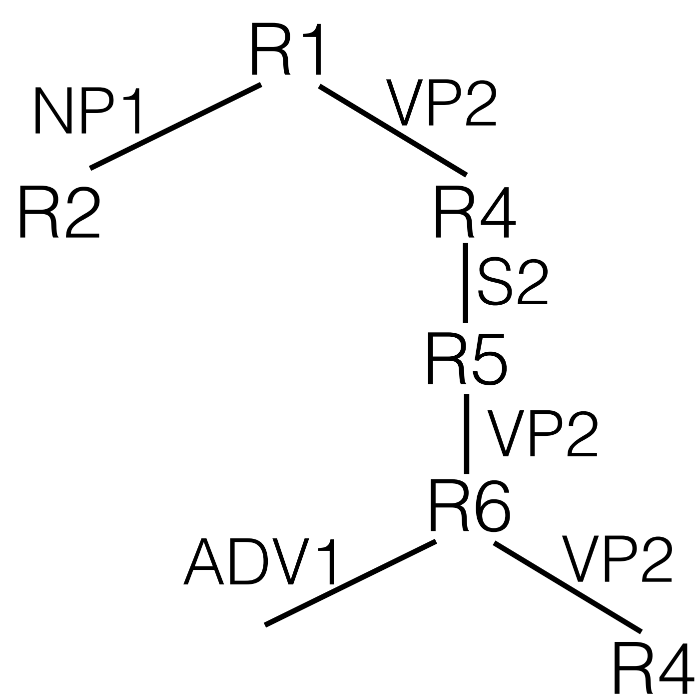


Derived Graph

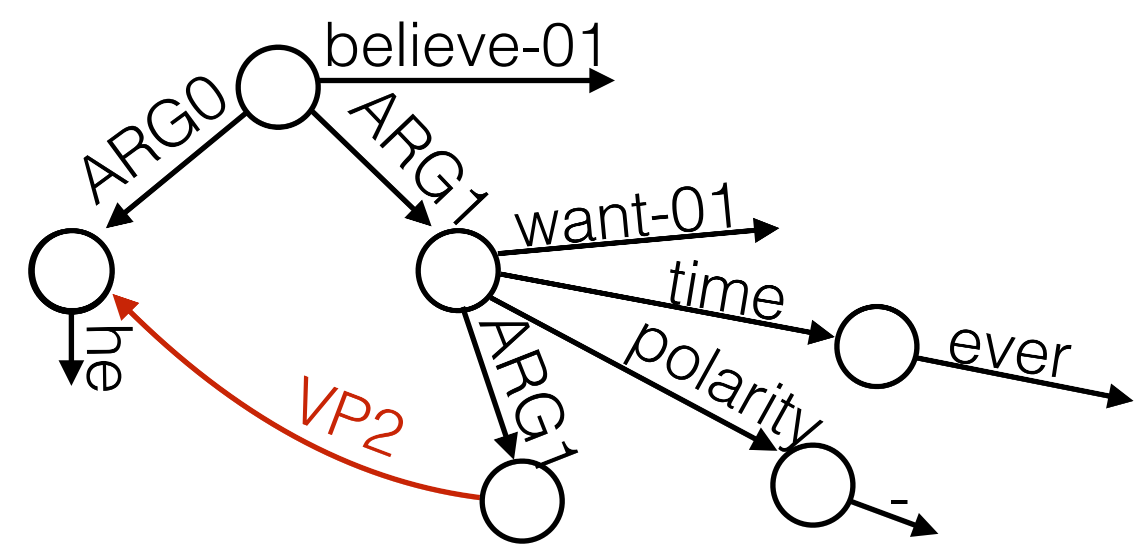


Hyperedge Replacement Grammar (HRG)

Derivation Tree

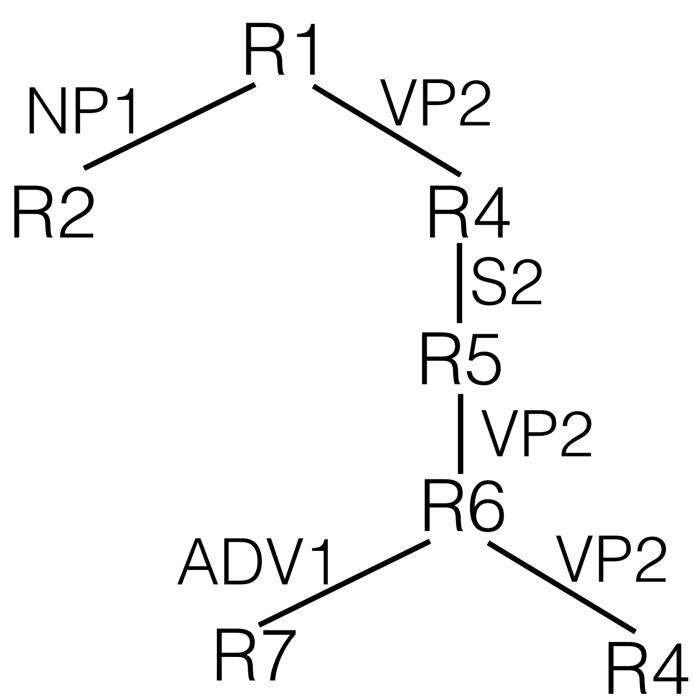


Derived Graph

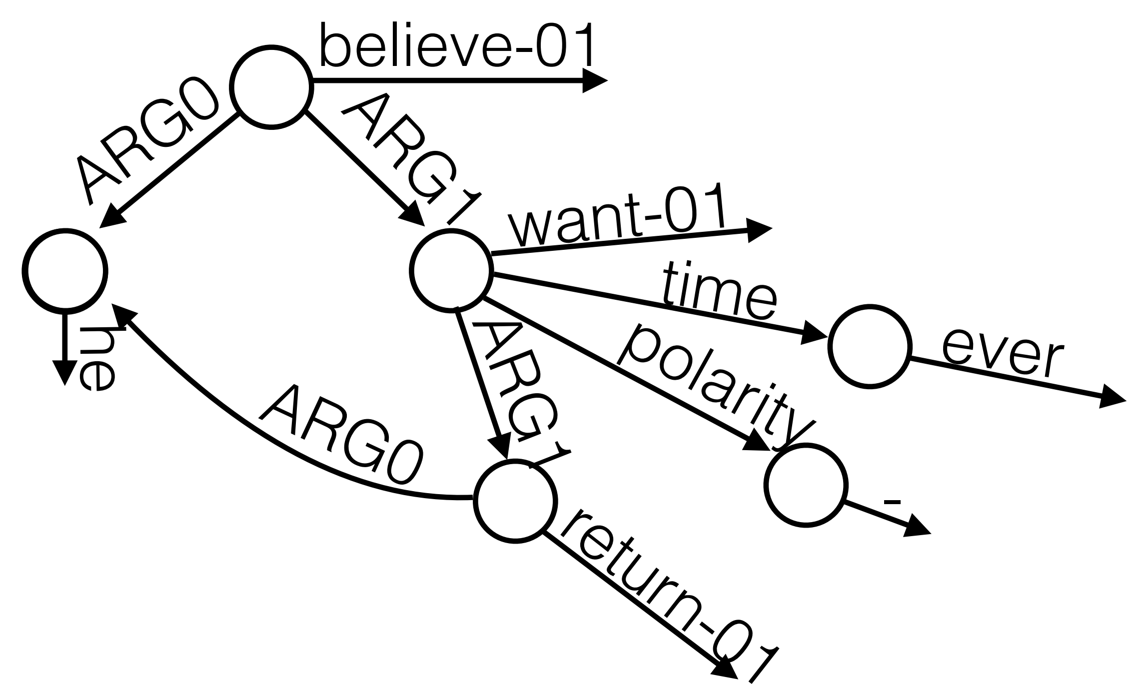


Hyperedge Replacement Grammar (HRG)

Derivation Tree

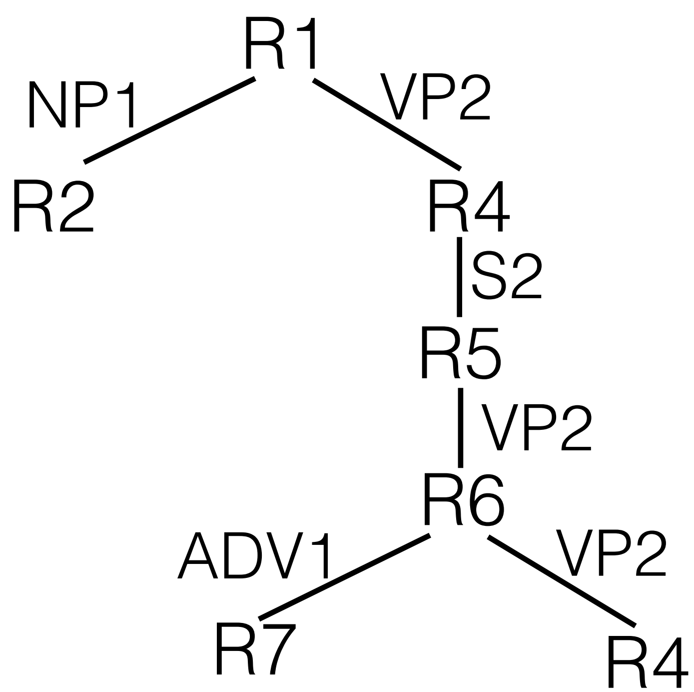


Derived Graph

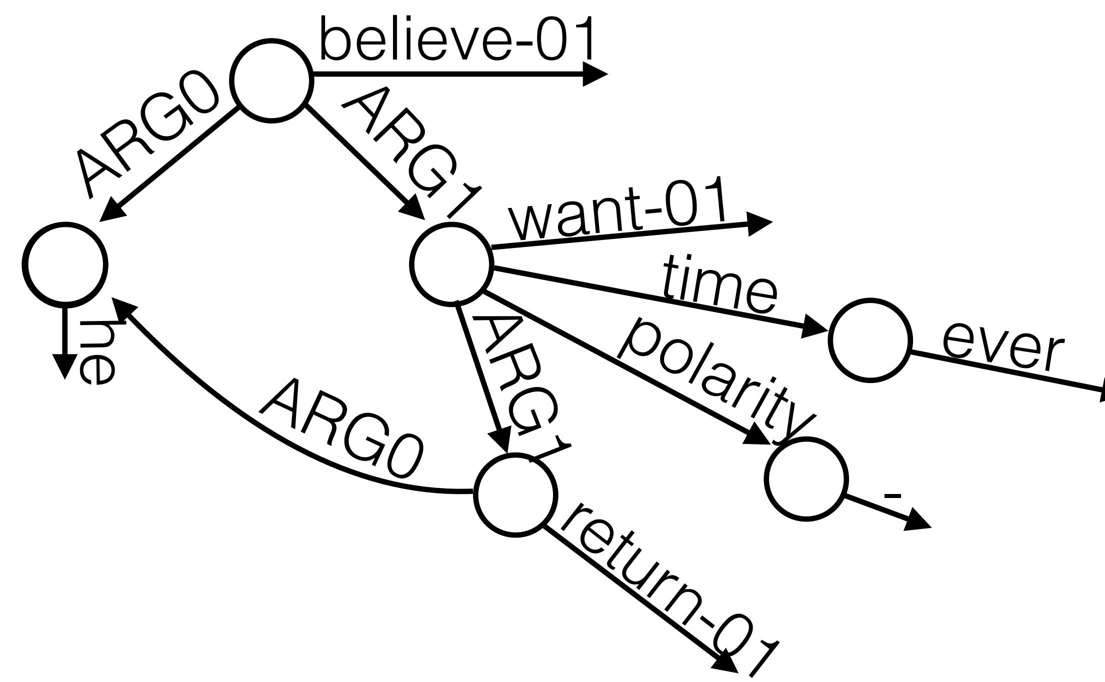


Hyperedge Replacement Grammar (HRG)

Derivation Tree

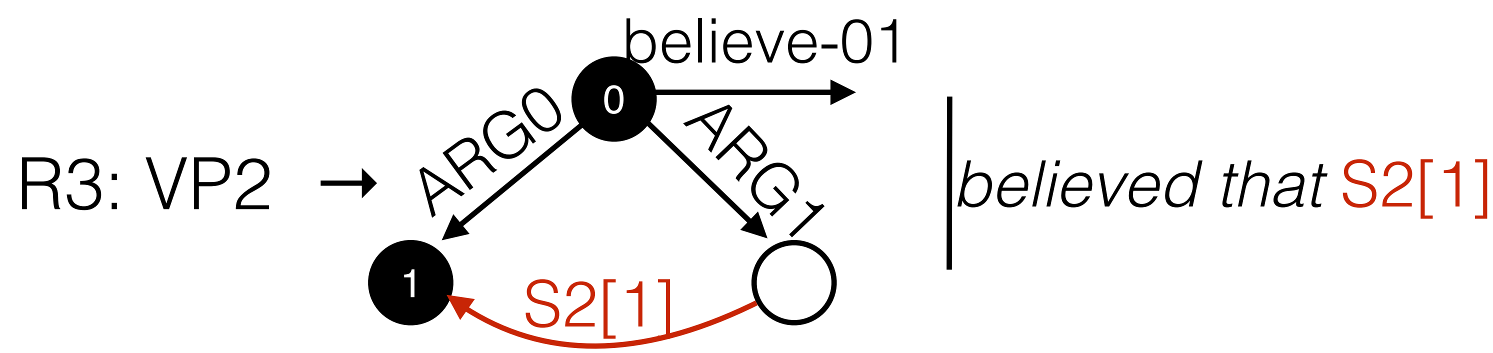


Derived Graph



- Polynomial time graph parsing algorithms exist.
  (Chiang, Andreas, Bauer, Hermann, Jones & Knight, 2013)

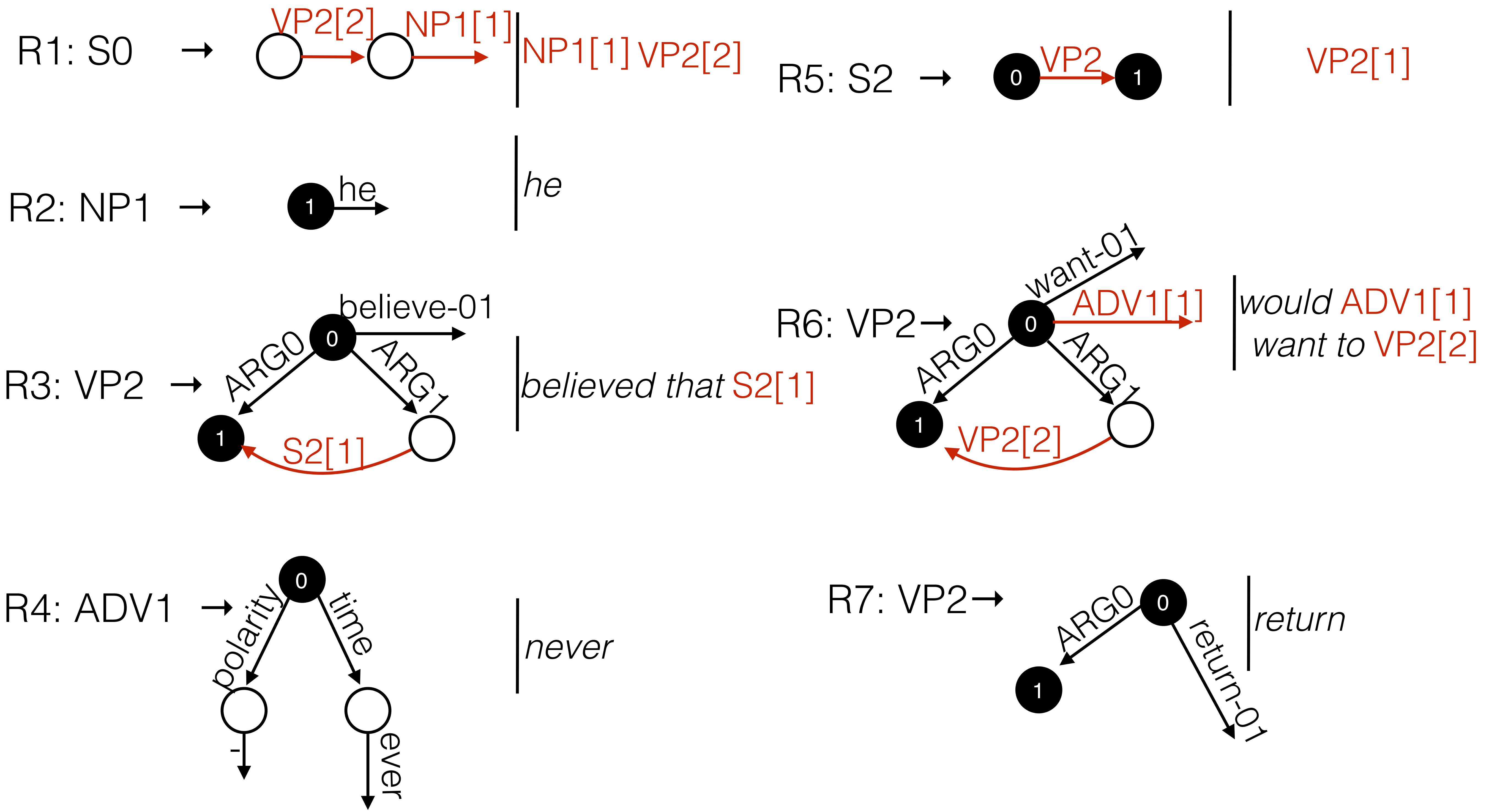
Synchronous Hyperedge Replacement Grammar (SHRG)



- string/graph sides share the same, typed nonterminal alphabet.
- Explicit synchronization (co-indexing).
- Every string derivation is also a valid graph derivation.
- SHRG derive string/graph **pairs.**

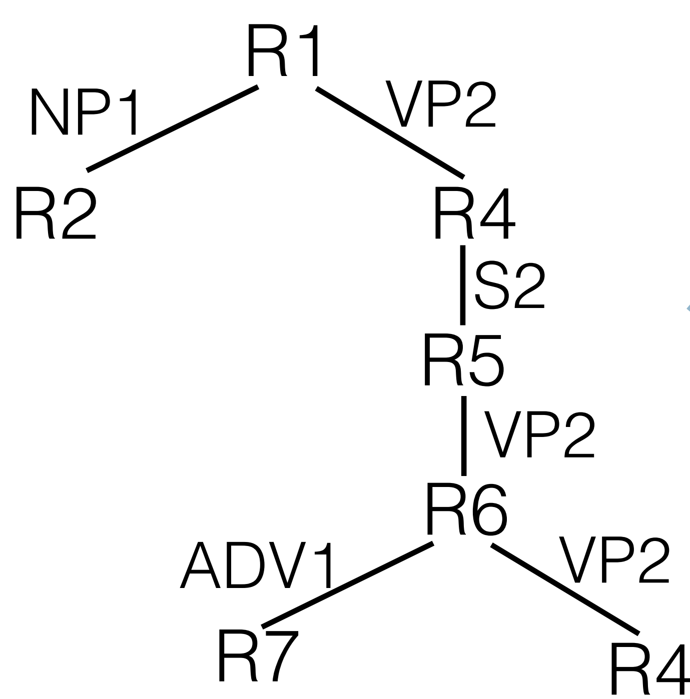
(Jones, Andreas, Bauer, Hermann & Knight, 2012)

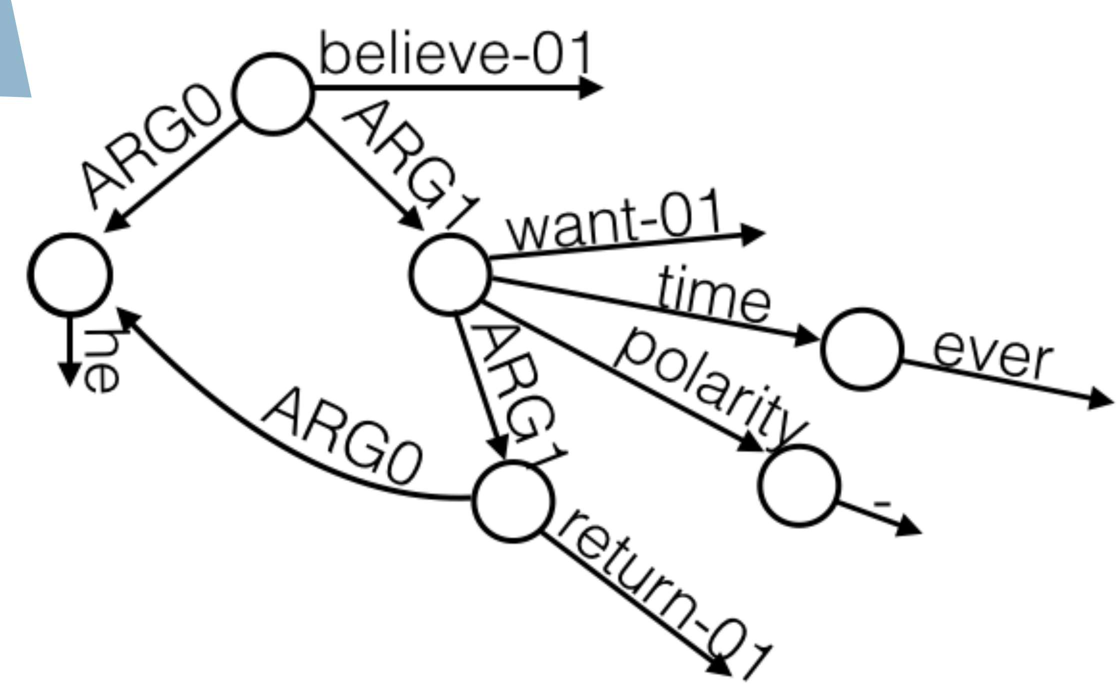
Synchronous Hyperedge Replacement Grammar (SHRG)



Synchronous Hyperedge Replacement Grammar (SHRG)

he believed that he would never want to return





SHRG for Semantic Construction

SHRG for Semantic Construction

- How does SHRG compare to other formalisms for semantic construction?

SHRG for Semantic Construction

- How does SHRG compare to other formalisms for semantic construction?
  - Less powerful than Feature Unification Grammars and CCG with λ-calculus.

SHRG for Semantic Construction

- How does SHRG compare to other formalisms for semantic construction?
  - Less powerful than Feature Unification Grammars and CCG with λ-calculus.
- Are there any limitations on the Graph Structures that SHRG can produce?

SHRG for Semantic Construction

- How does SHRG compare to other formalisms for semantic construction?
  - Less powerful than Feature Unification Grammars and CCG with λ-calculus.
- Are there any limitations on the Graph Structures that SHRG can produce?
  - Yes, restriction to CFG limits graph structures that can be constructed. For example, no cross-serial dependencies.

SHRG for Semantic Construction

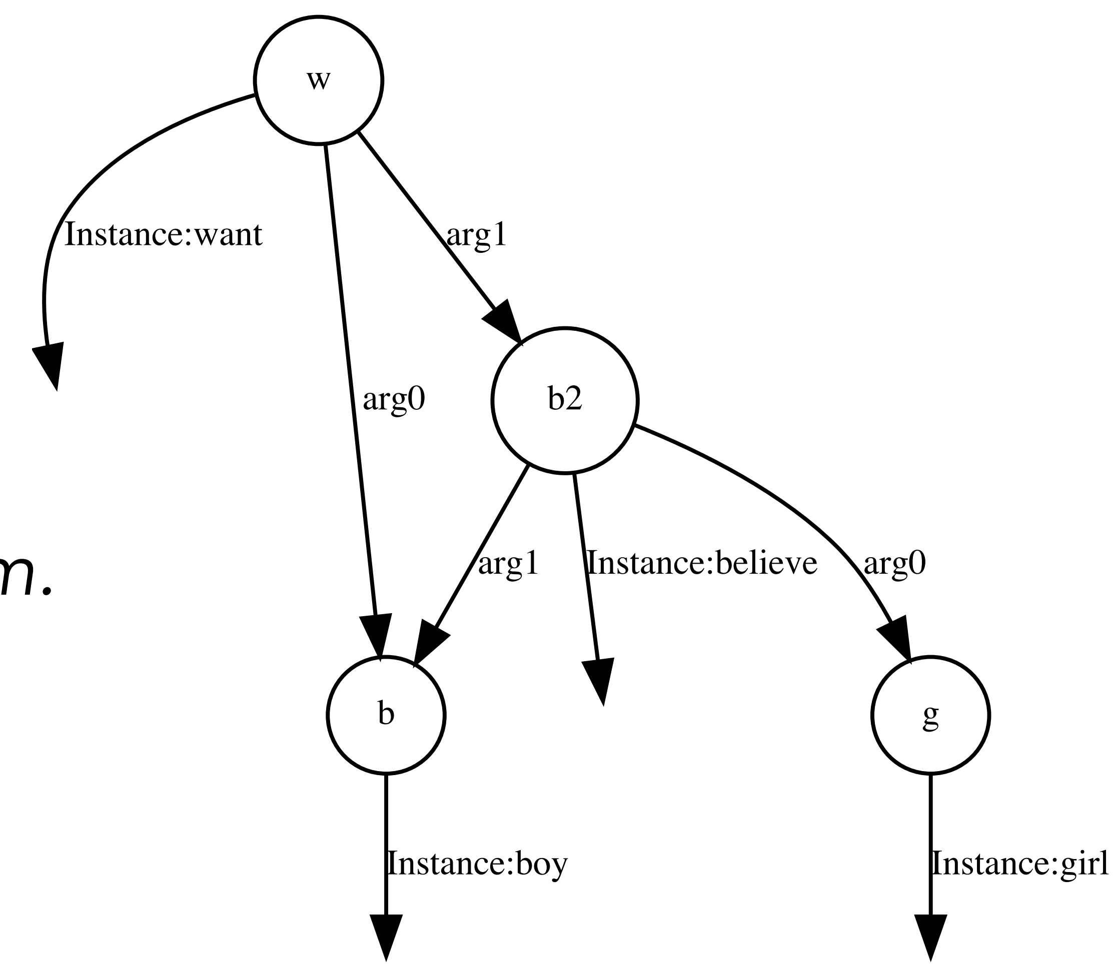
- How does SHRG compare to other formalisms for semantic construction?
  - Less powerful than Feature Unification Grammars and CCG with λ-calculus.
- Are there any limitations on the Graph Structures that SHRG can produce?
  - Yes, restriction to CFG limits graph structures that can be constructed. For example, no cross-serial dependencies.
  - Yes, maximum hyperedge type in a grammar implies a hierarchy of grammars. Cannot build up arbitrary complexity.

Extracting Synchronous Grammars

Task: Given a corpus of string/graph pairs, learn a SHRG that

- can derive all string/graph pairs in the corpus.
- is compact (small number of rules).
- generalizes well to unseen data.

The boy wants the girl to believe him.

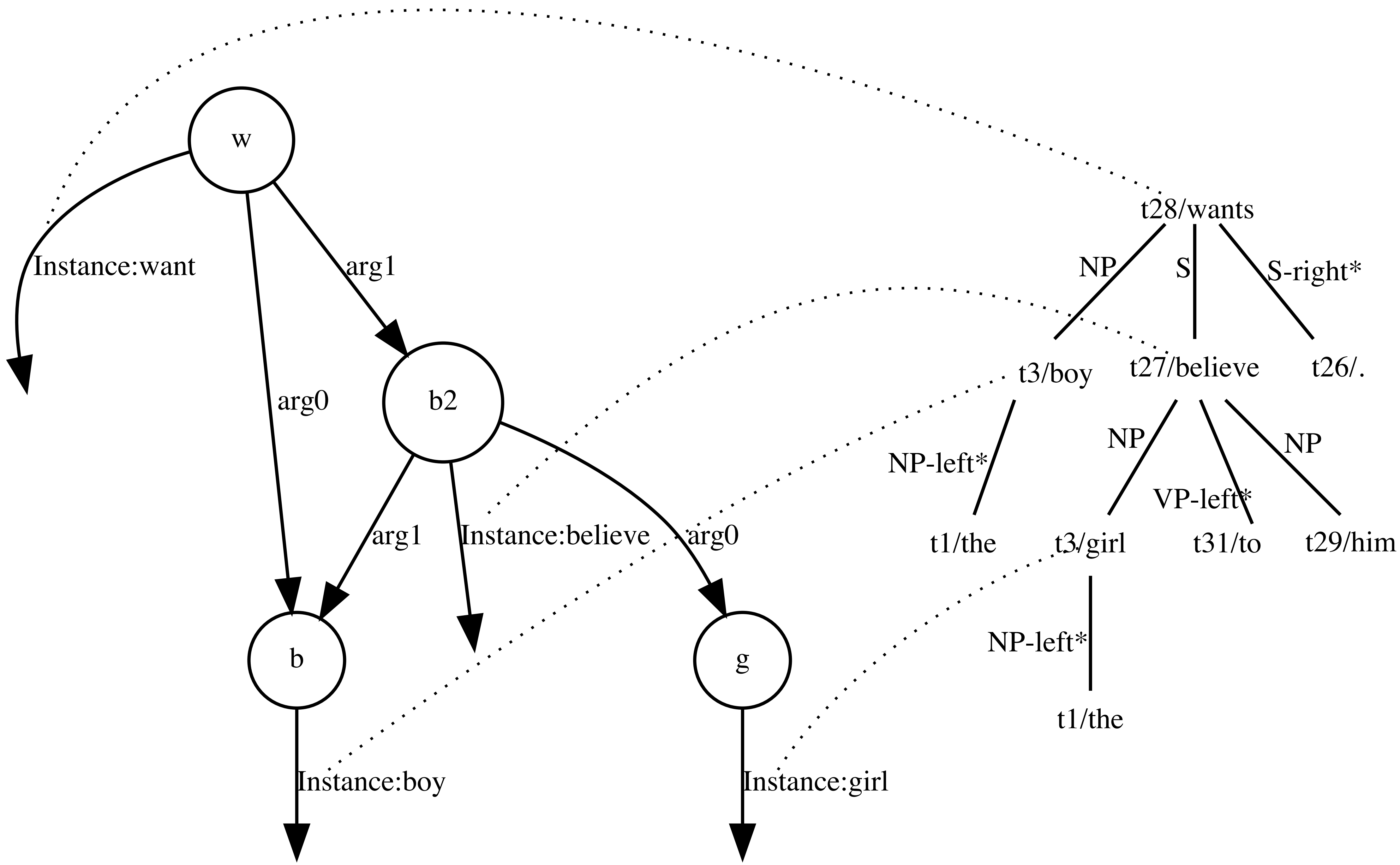


Extracting SHRG/LSHRG Grammar Rules

My approach uses syntactic derivations to guide rule extraction.

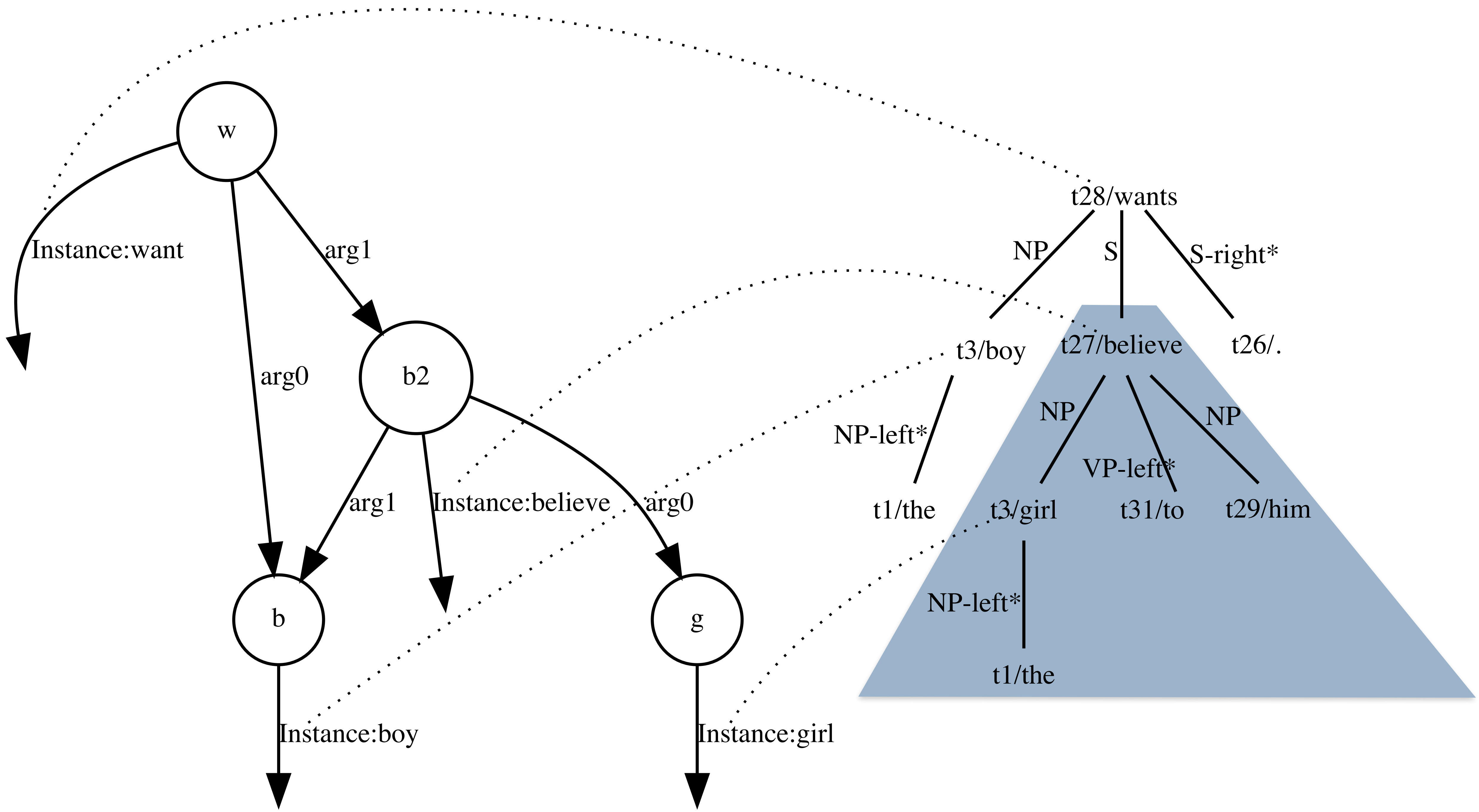
1. Syntactic parser creates string derivations (MICA parser).
2. Aligner computes string-to-graph alignments.
3. Partitioning algorithm creates a rule forest.
4. EM-based rule selection extracts a compact set of rules.

Alignments And Derivation Trees



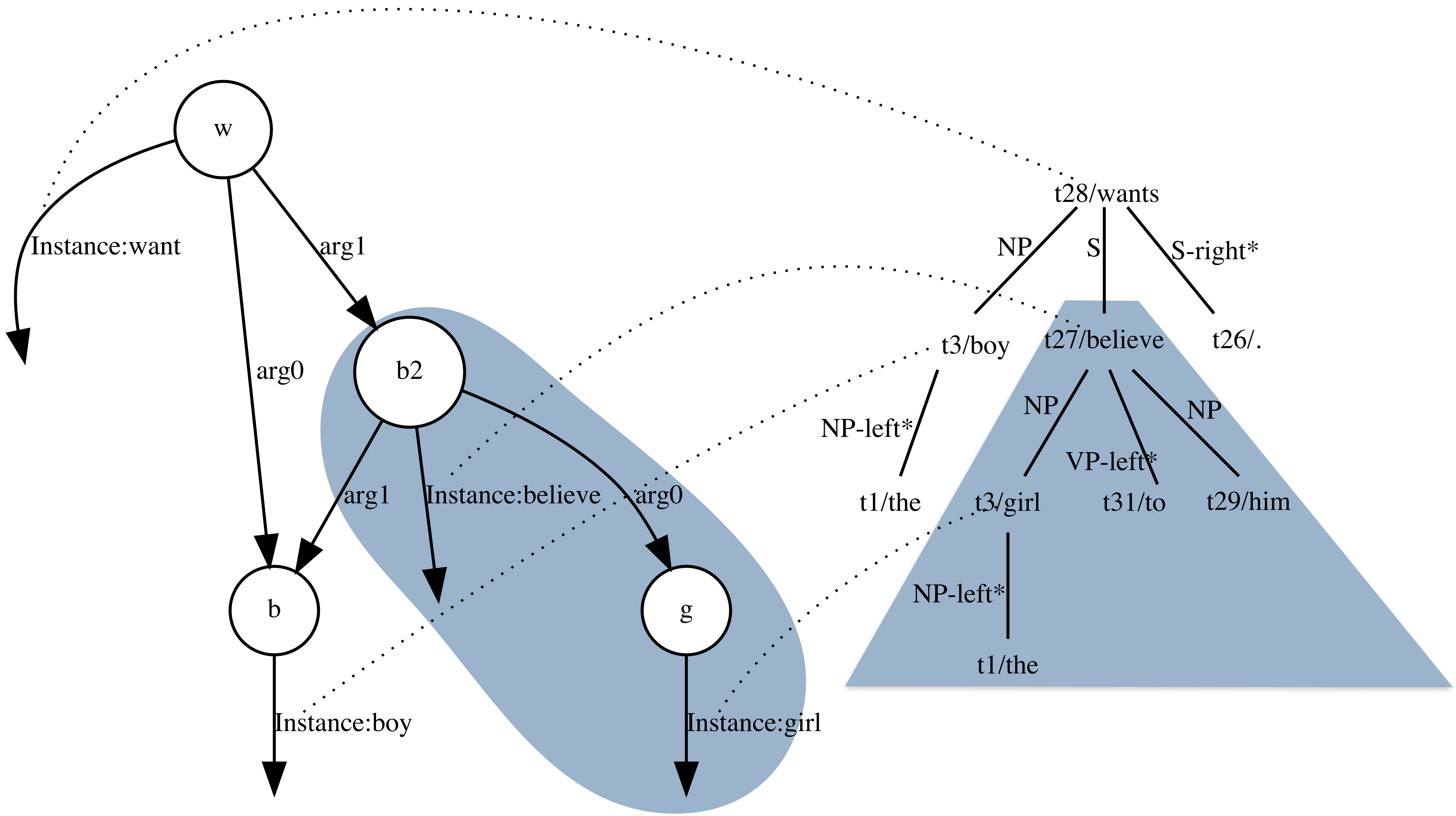
- Project alignments to nodes in the derivation tree.
- Every subtree is aligned to a subset of graph edges.

Alignments And Derivation Trees



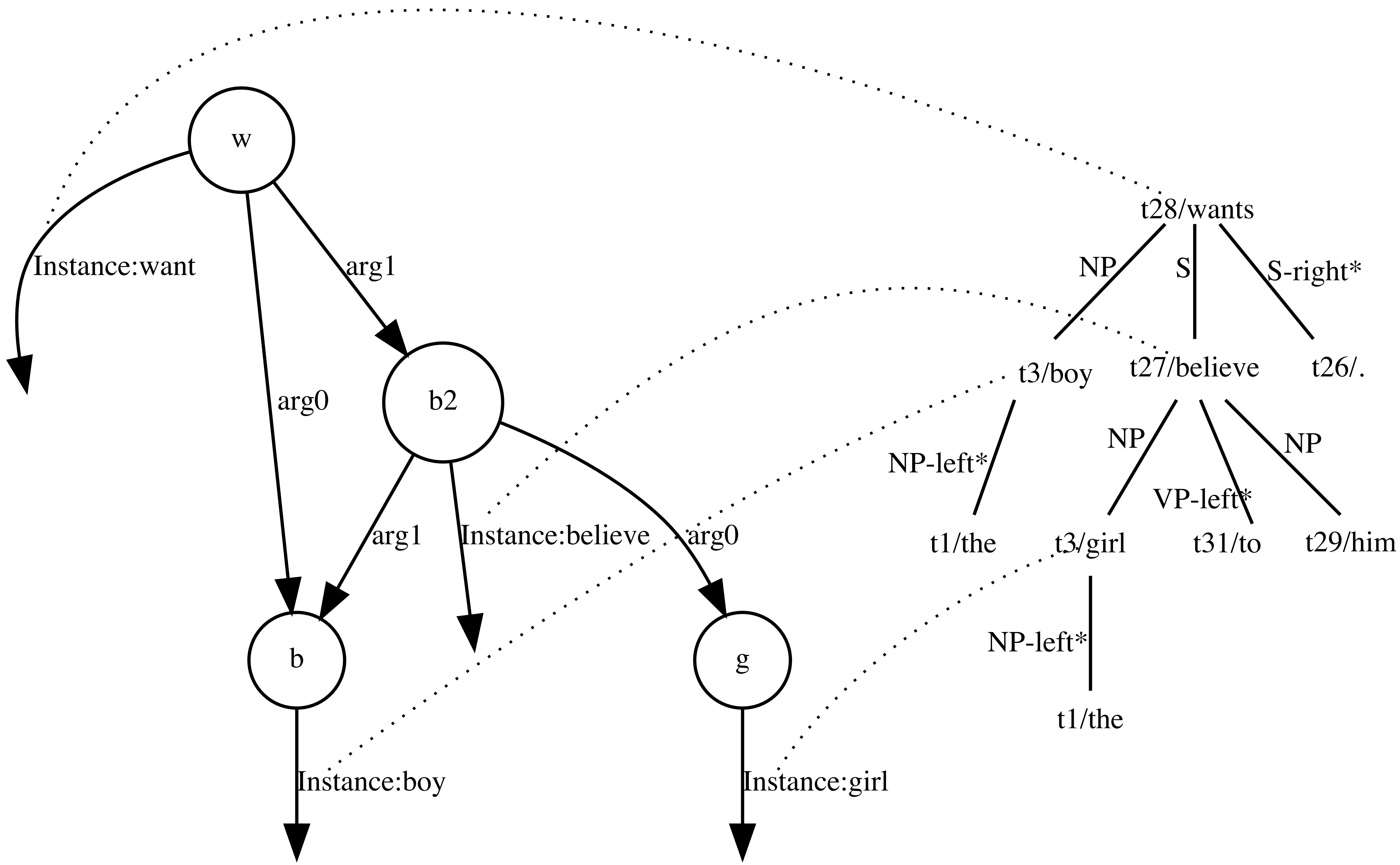
- Project alignments to nodes in the derivation tree.
- Every subtree is aligned to a subset of graph edges.

Alignments And Derivation Trees

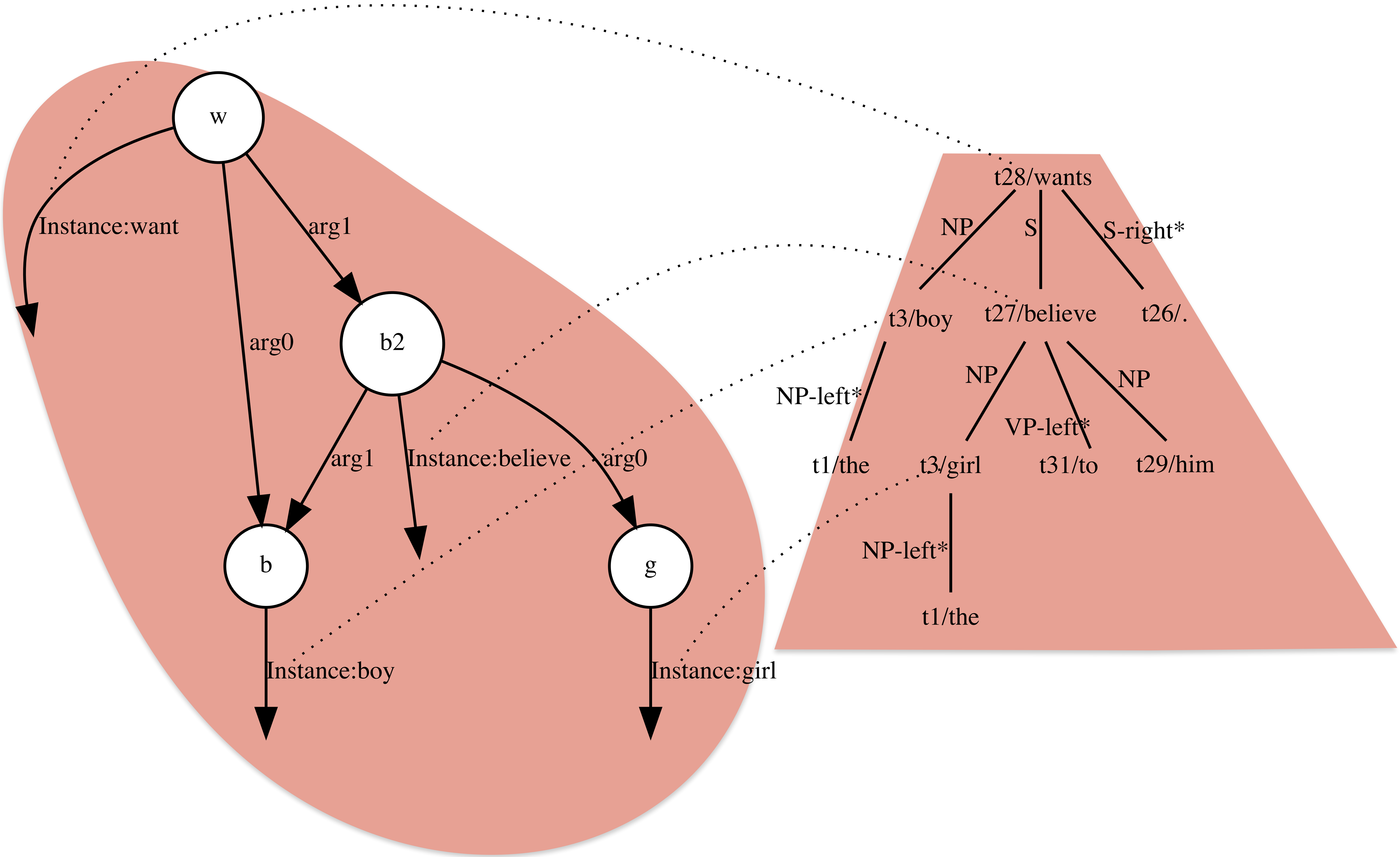


- Project alignments to nodes in the derivation tree.
- Every subtree is aligned to a subset of graph edges.

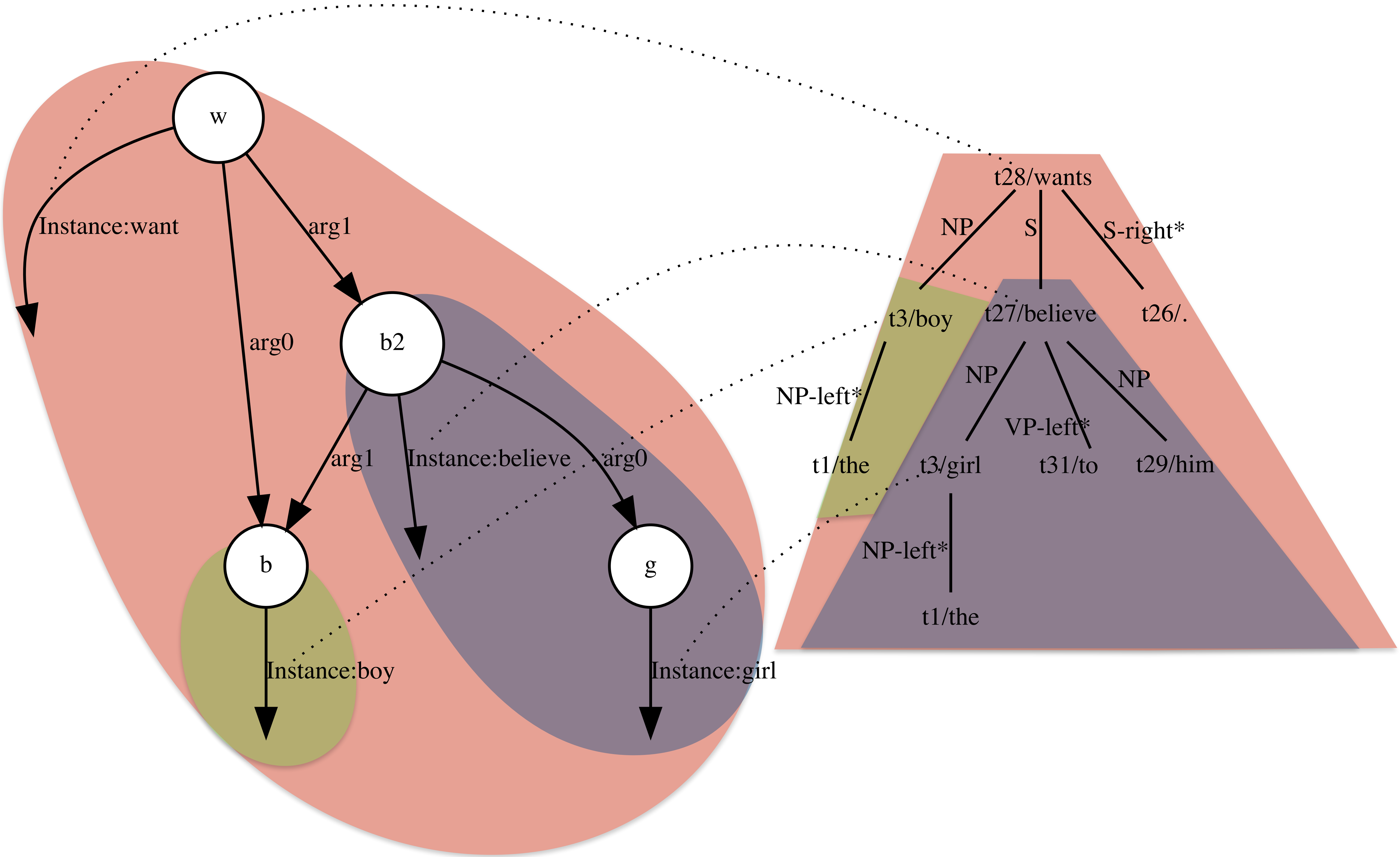
Partitioning Algorithm



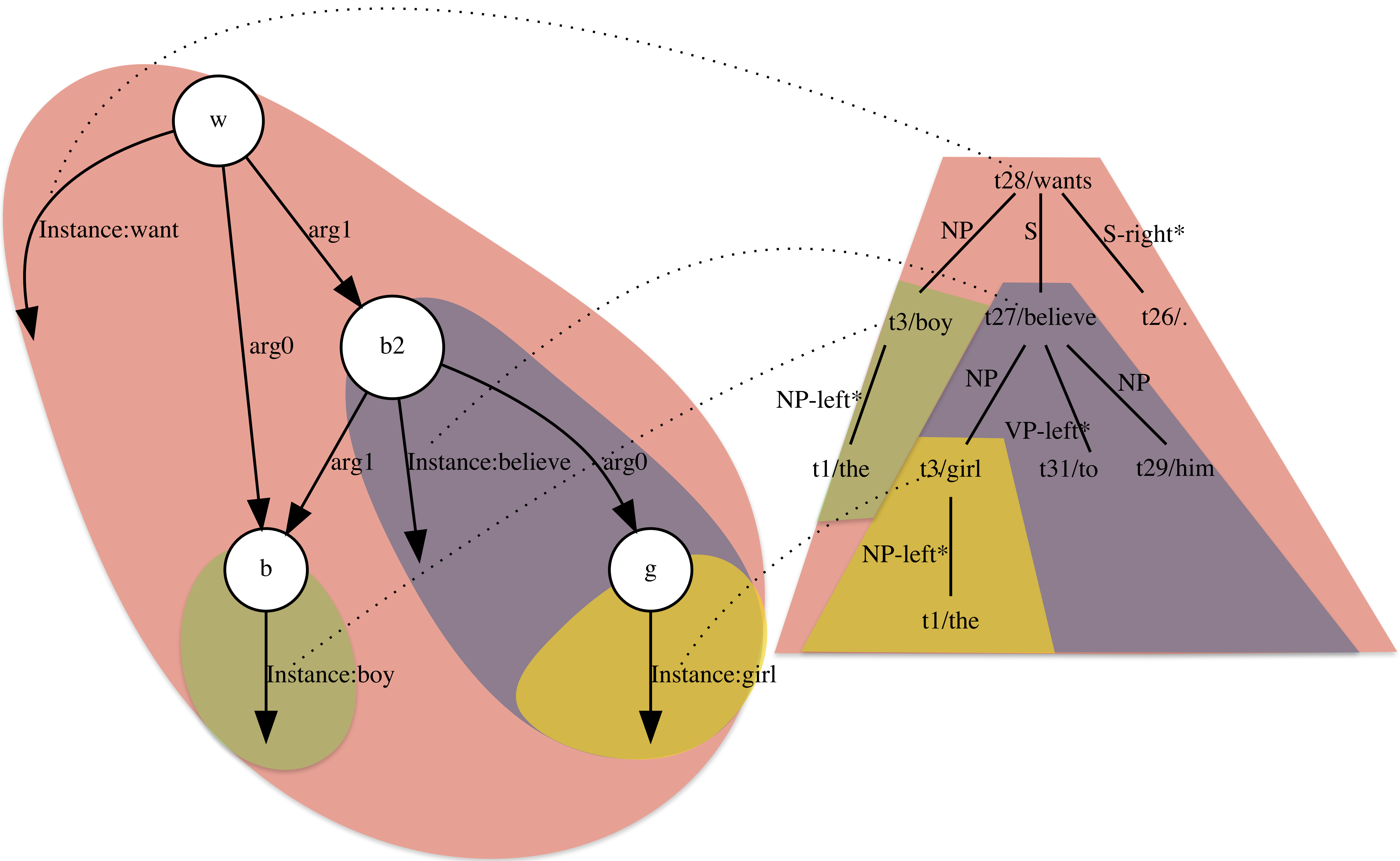
Partitioning Algorithm



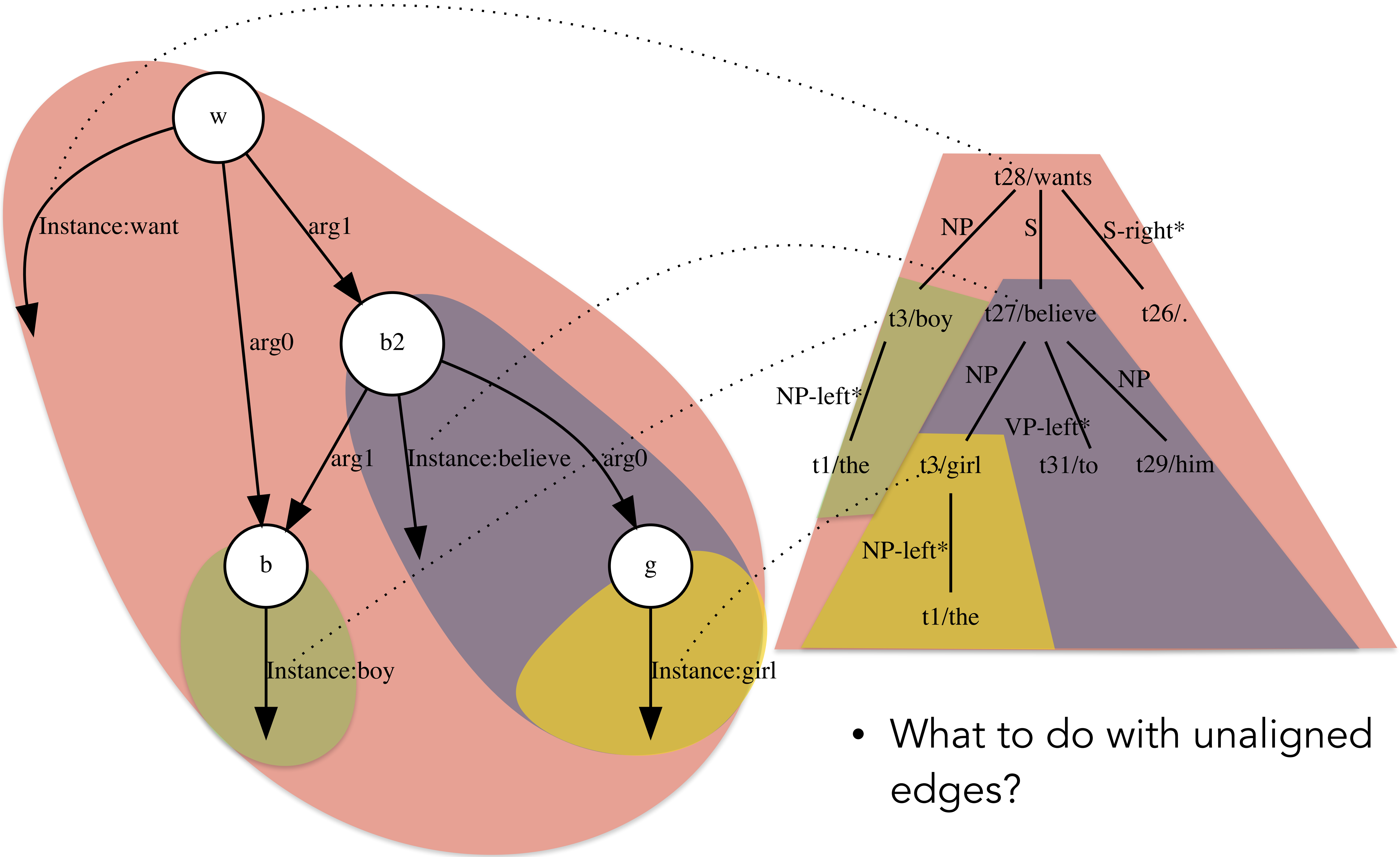
Partitioning Algorithm



Partitioning Algorithm

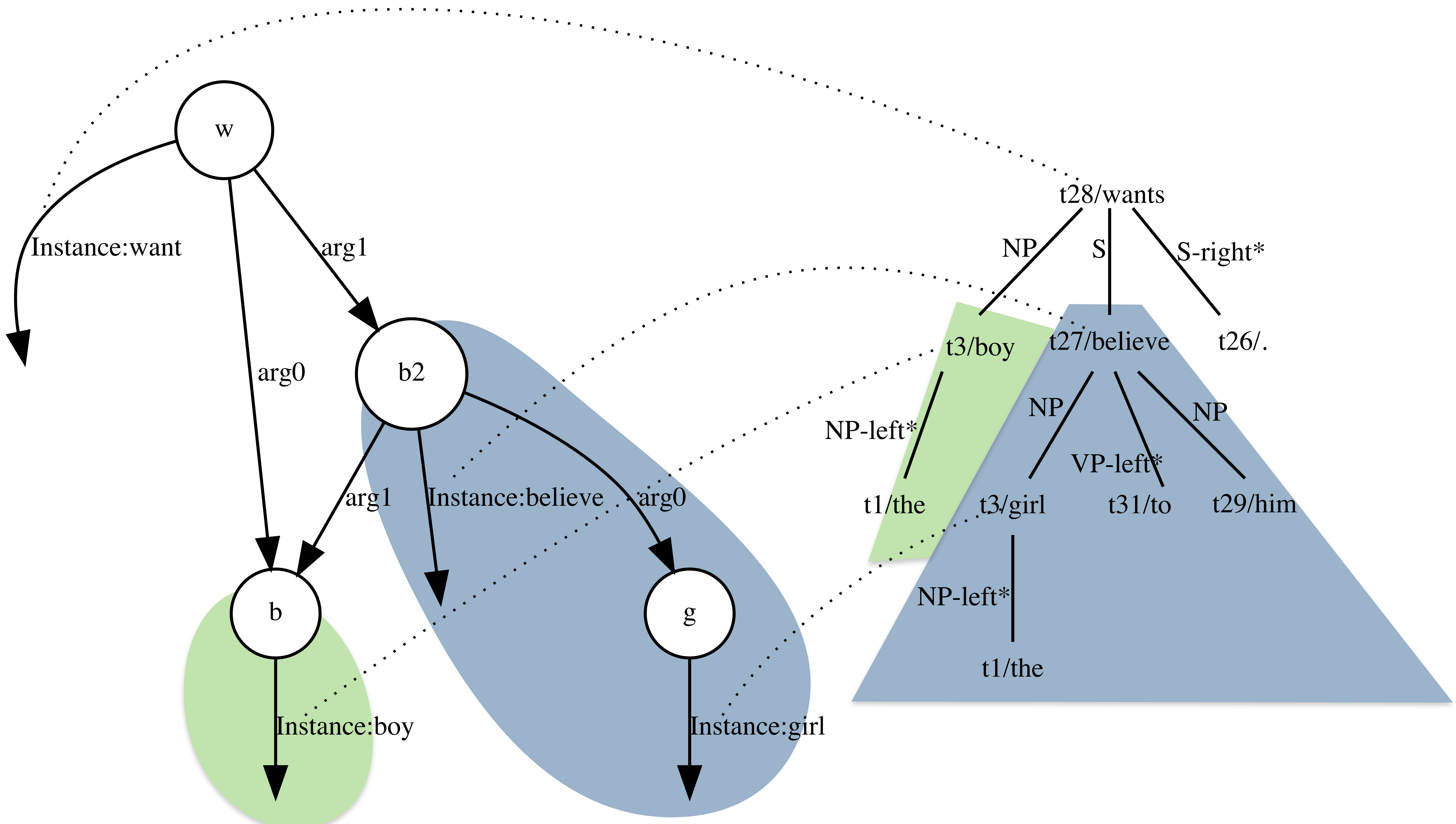


Partitioning Algorithm

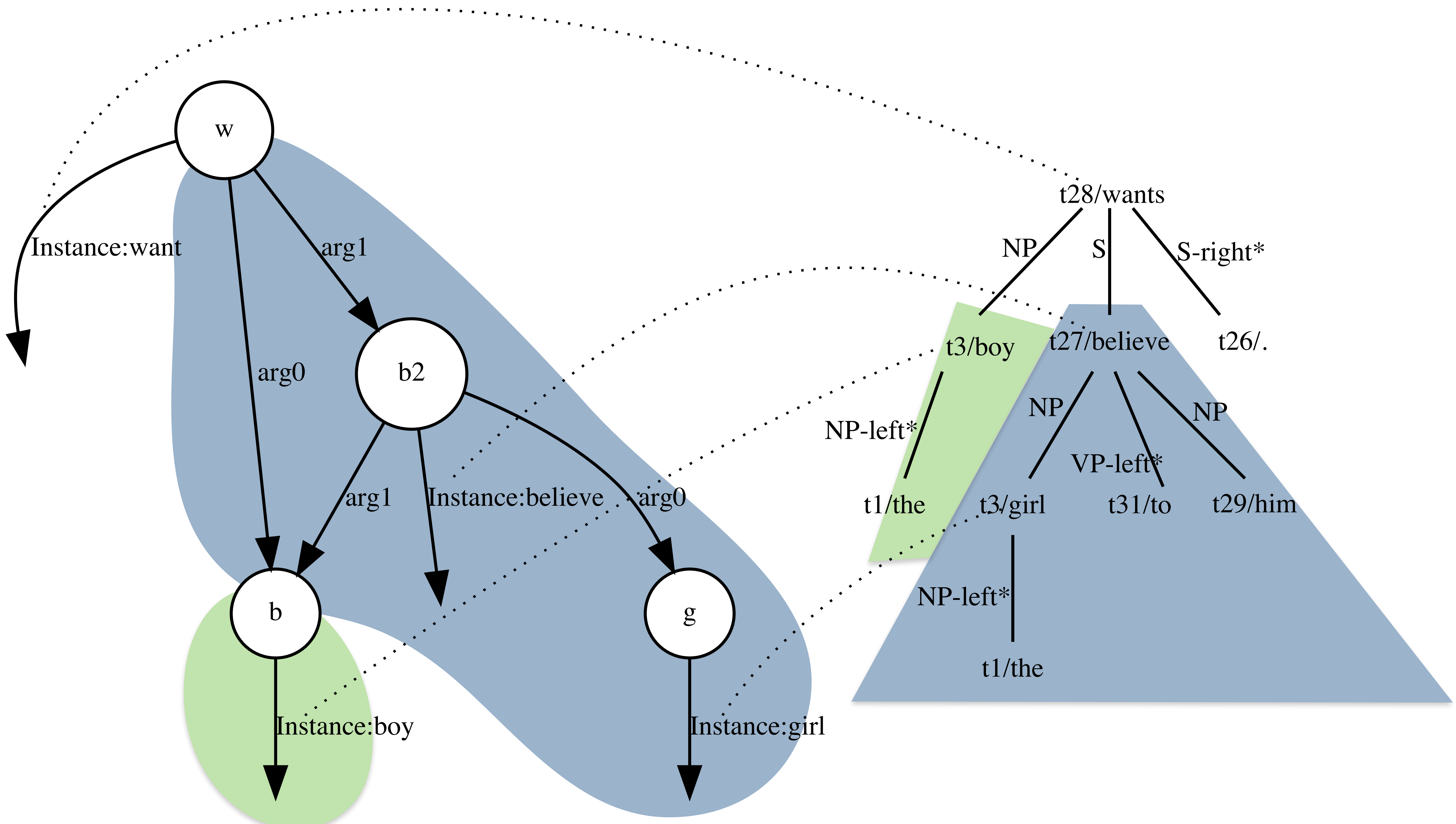


- What to do with unaligned edges?

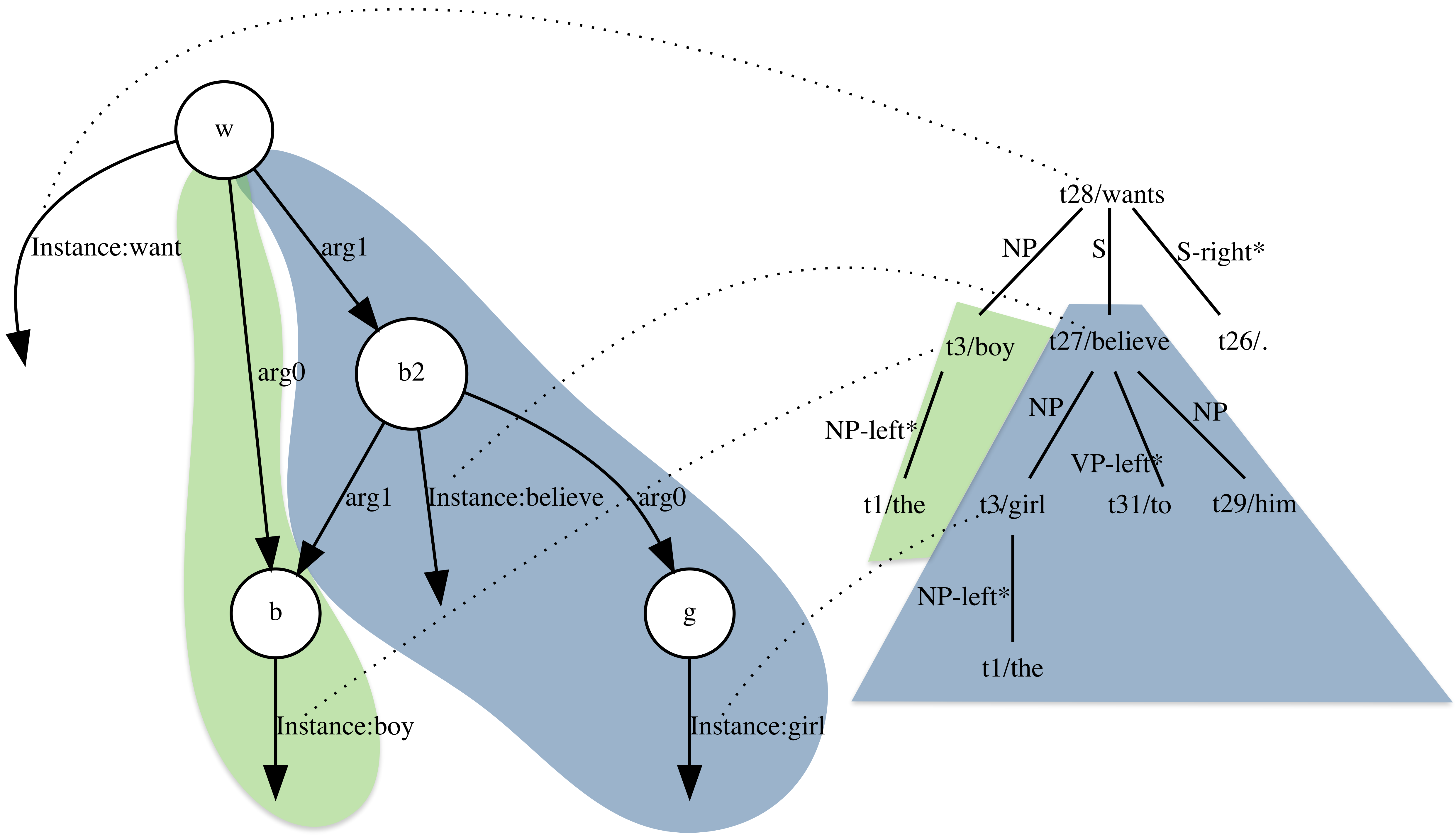
Binary Partitioning



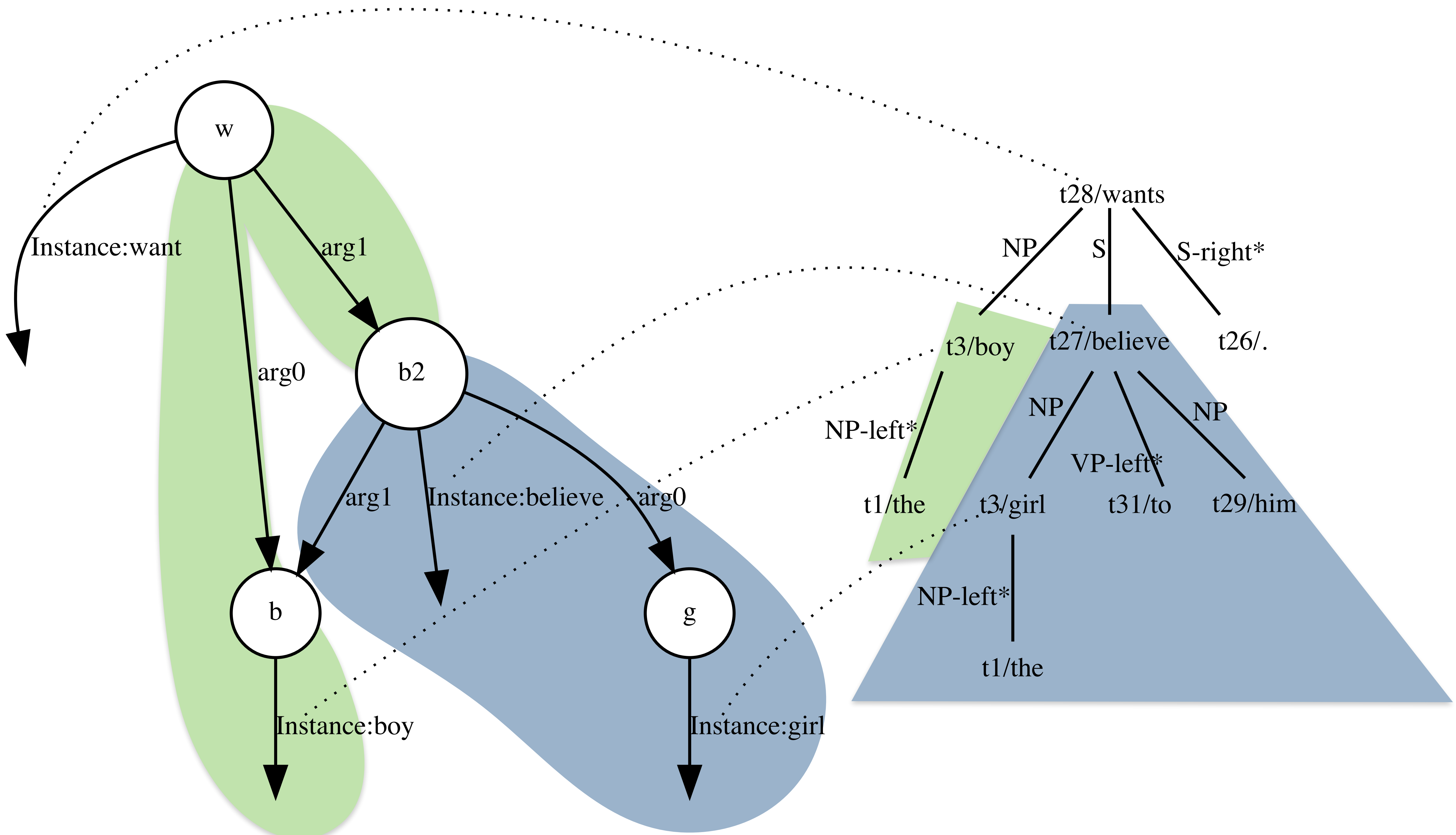
Binary Partitioning



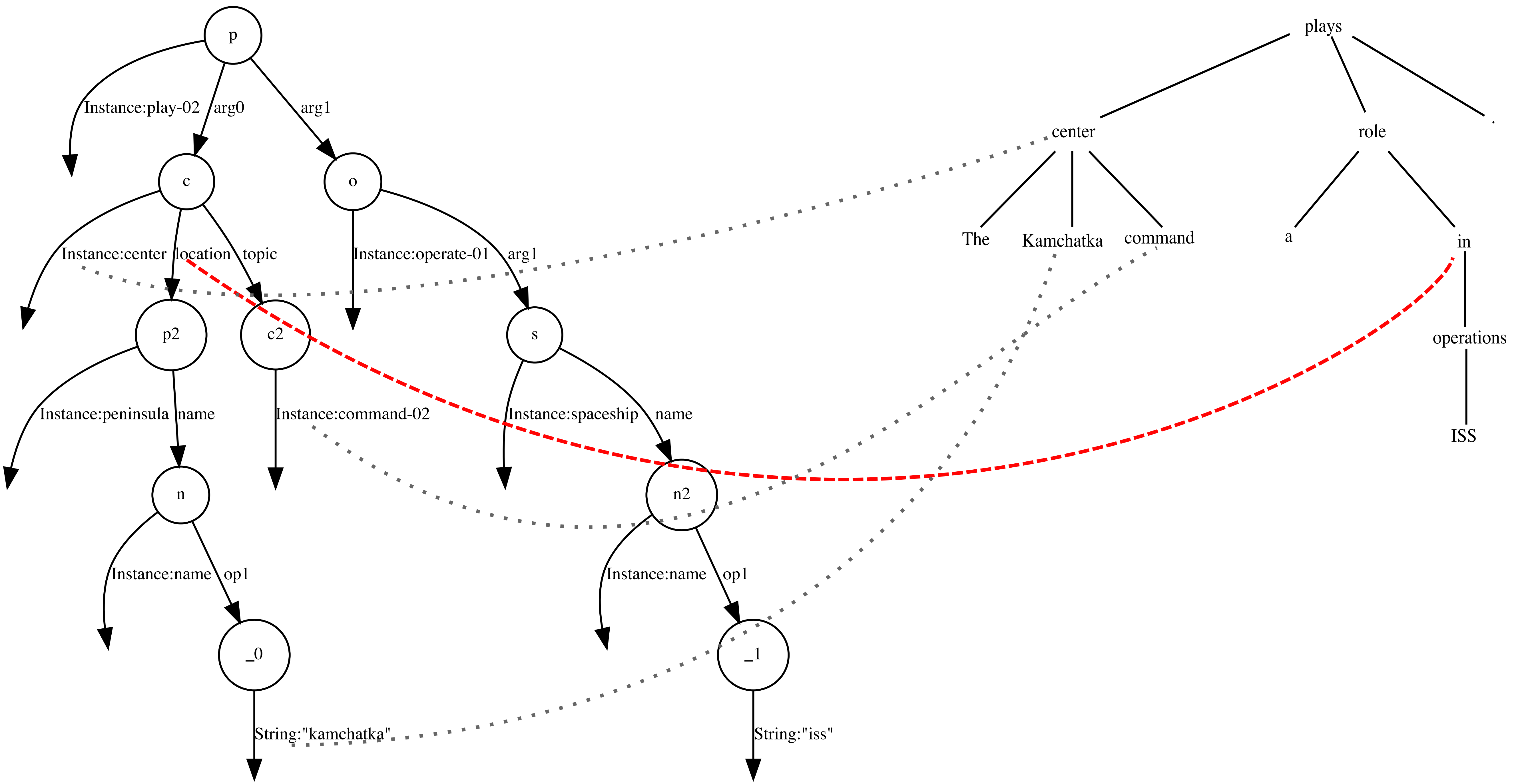
Binary Partitioning



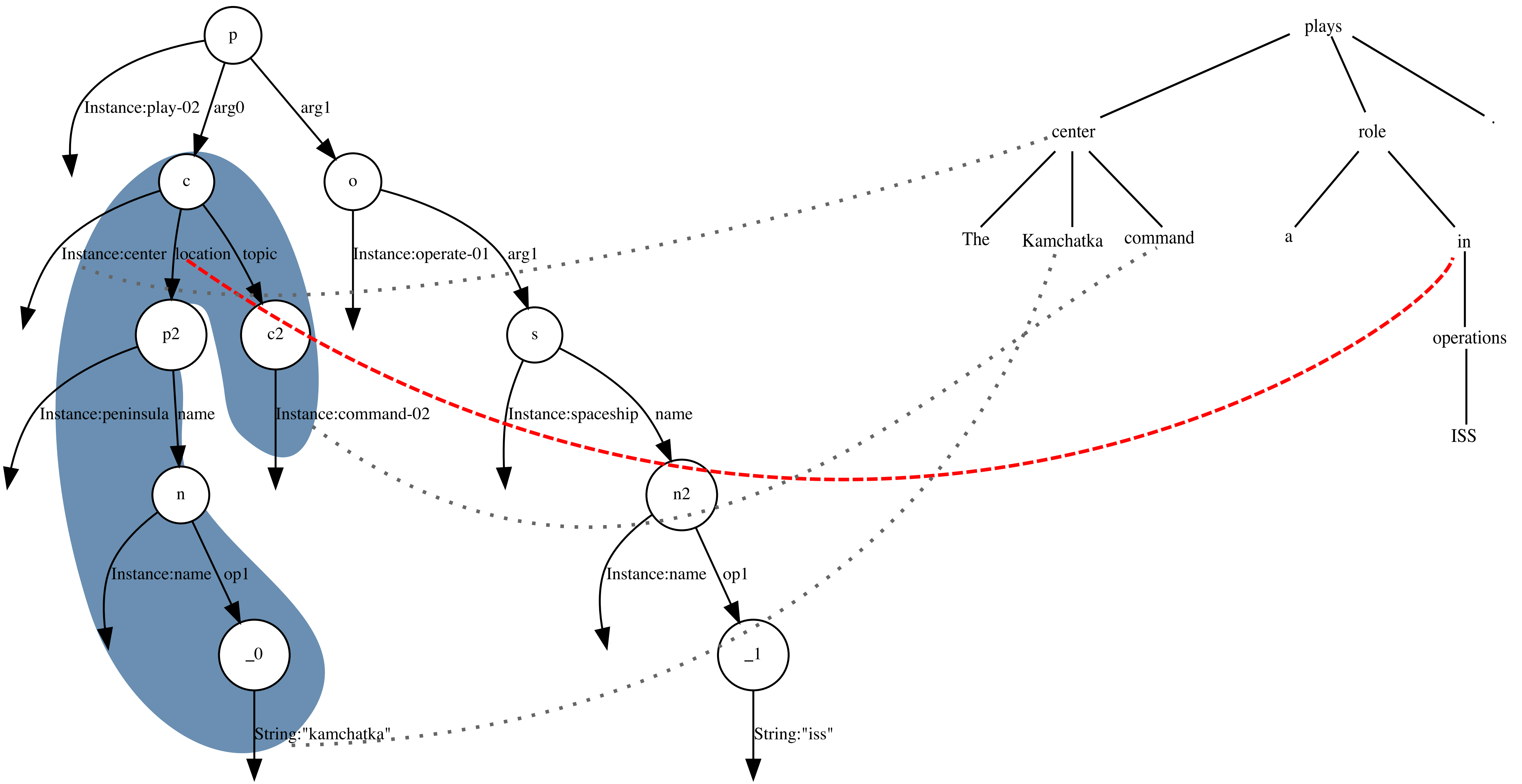
Binary Partitioning



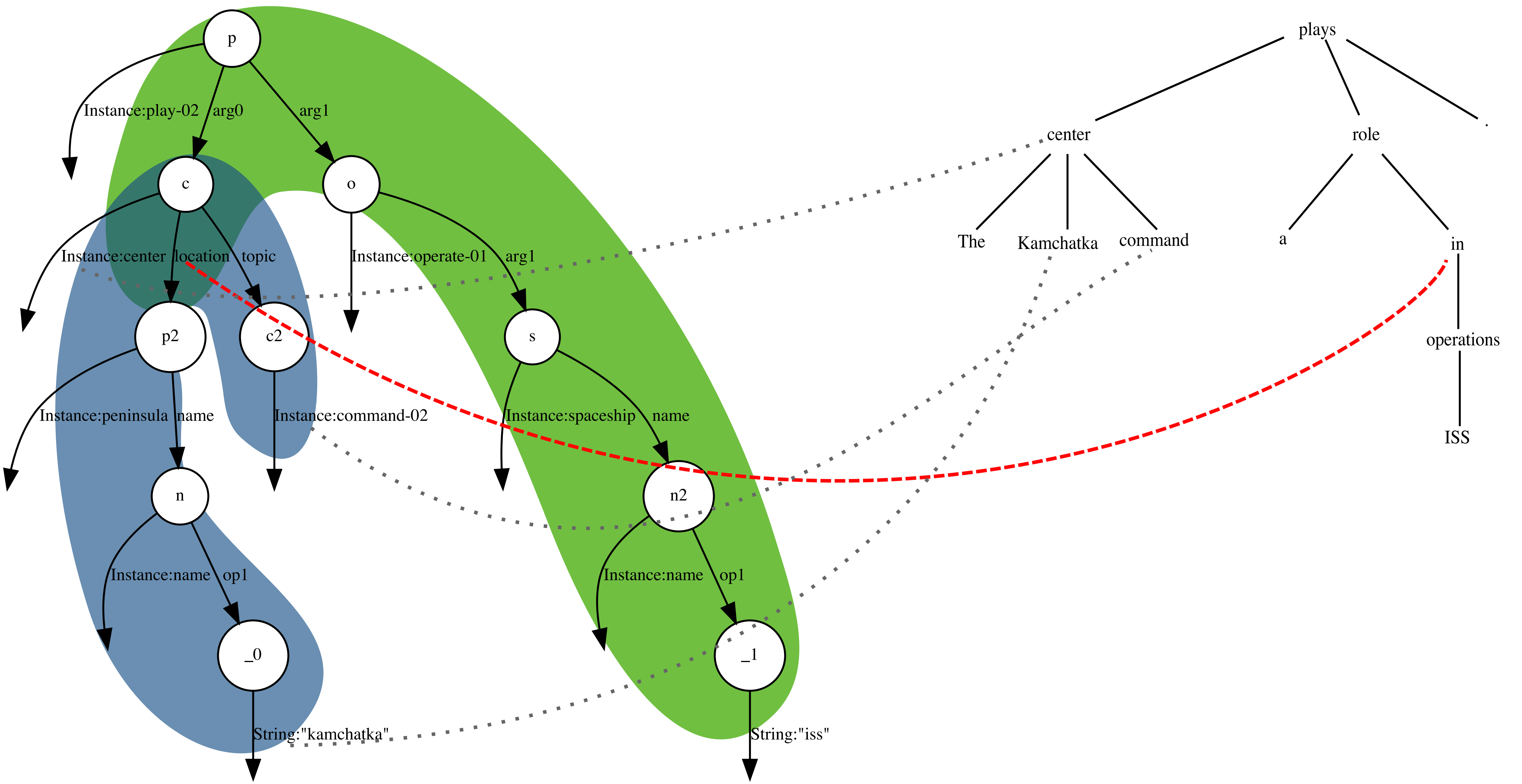
Problematic Alignments



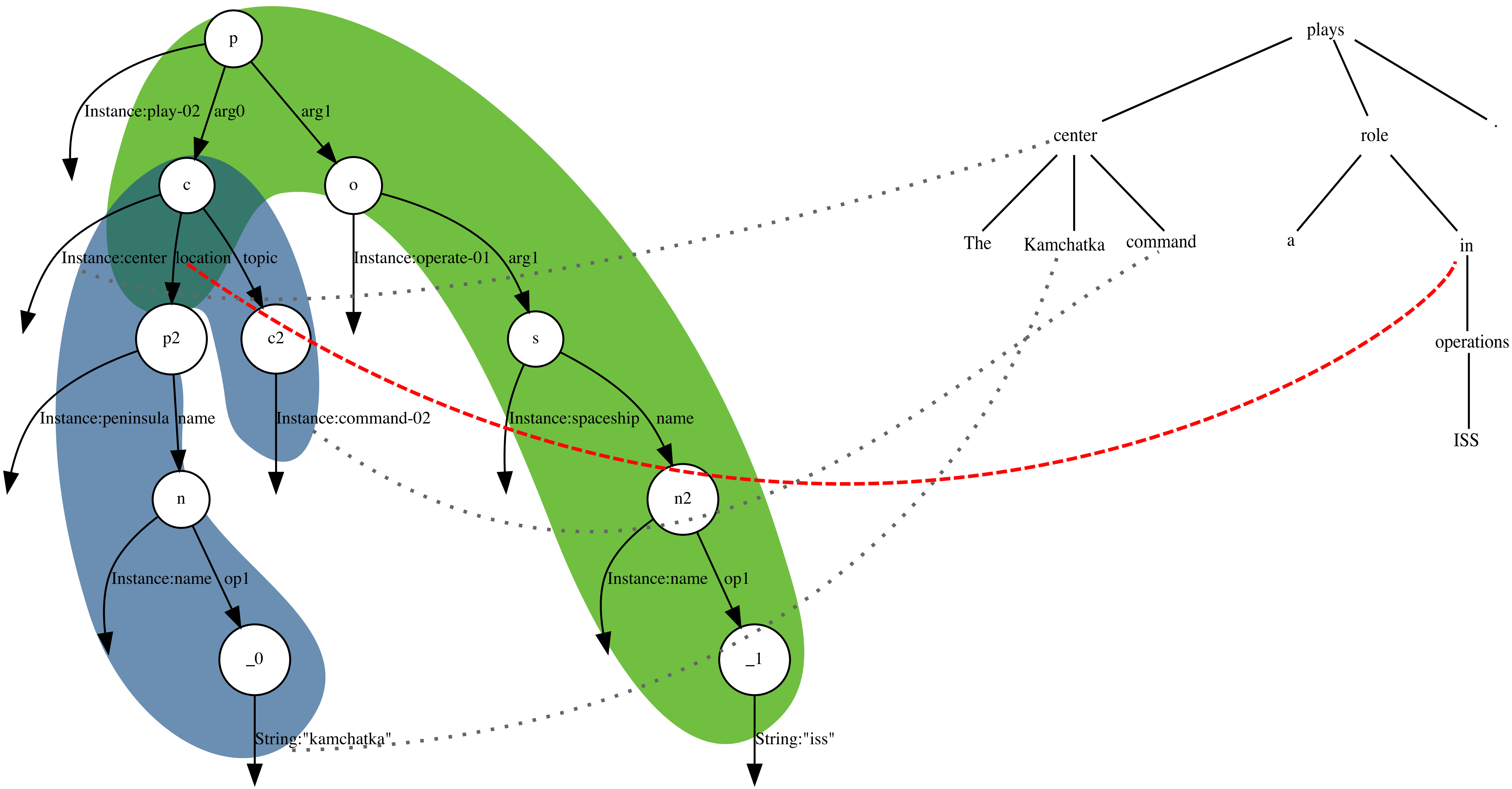
Problematic Alignments



Problematic Alignments



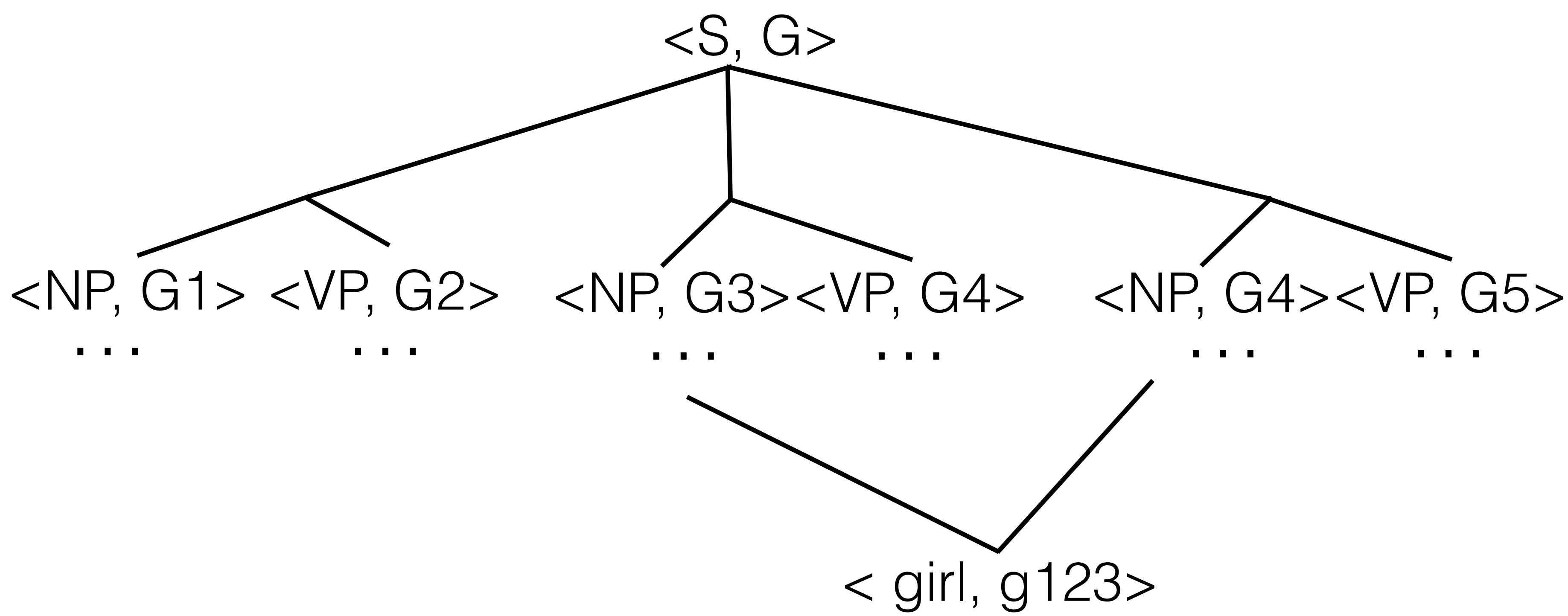
Problematic Alignments



Solution: Compute spanning trees for each phrase.
Remove edge alignments in the intersection.

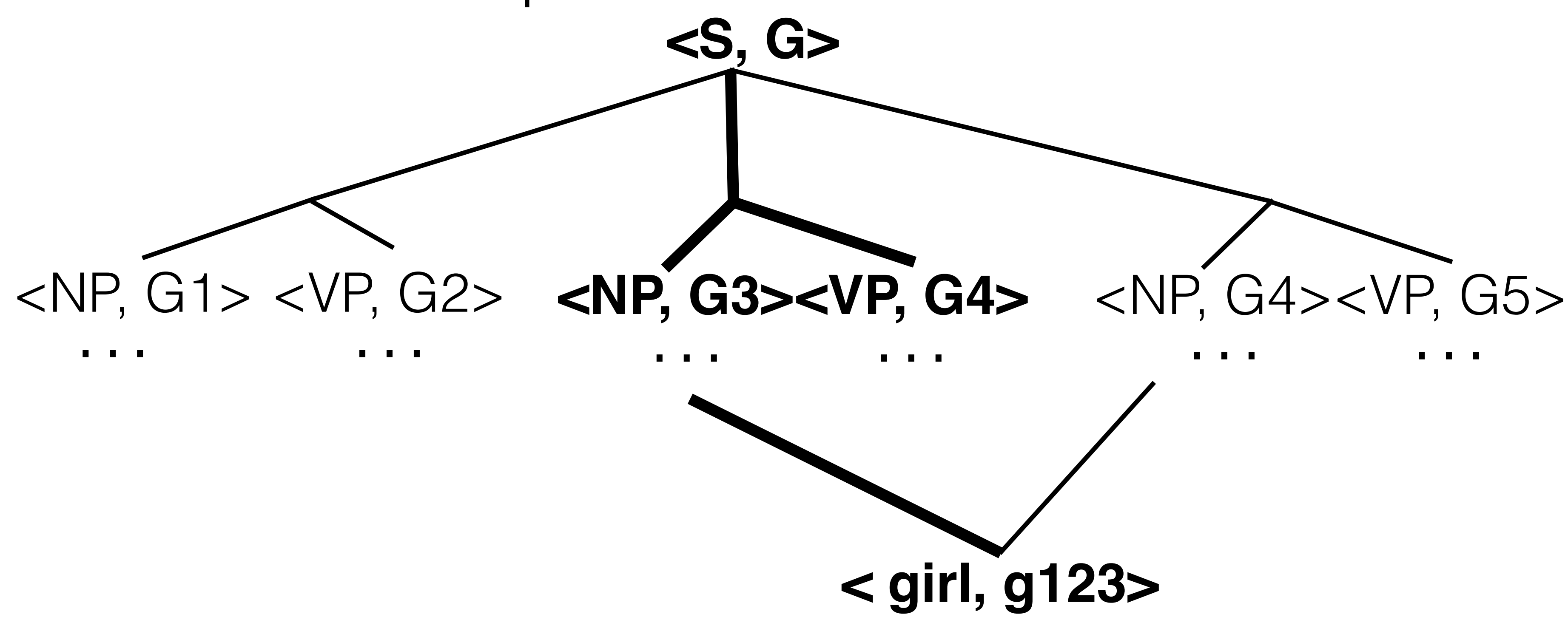
Constructing A Decomposition Forest

- For each way of partitioning the graph, recursively run partitioning on the subtrees and subgraphs.
- This is expensive! Need memoization.
- The result is a packed forest.



Selecting a Grammar

- Extract forests for all string/graph pairs in the training data
- Run EM with inside/outside algorithm on the collection of forests.
- Select 1st best Viterbi tree from each forest. Extract rules from these forests and keep trees as “bronze derivations”



Selecting a Grammar

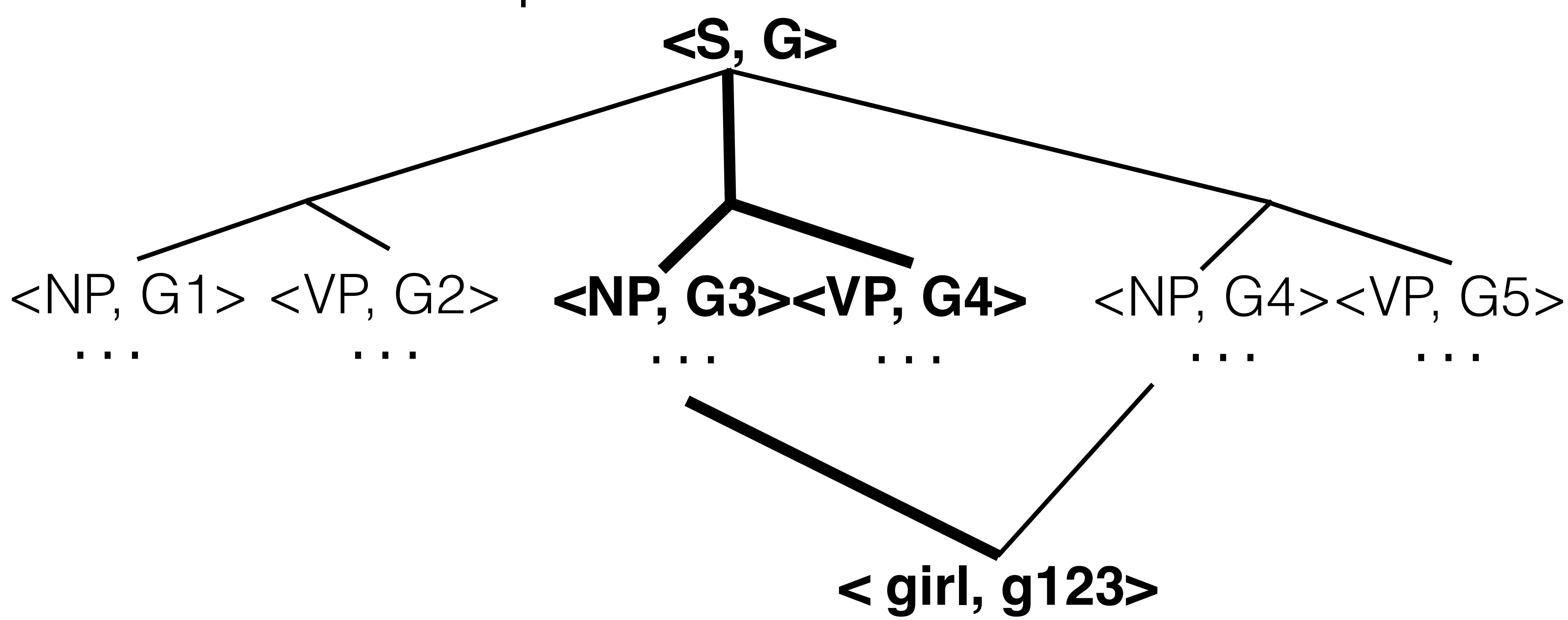
- Extract forests for all string/graph pairs in the training data

417k rules

- Run EM with inside/outside algorithm on the collection of forests.

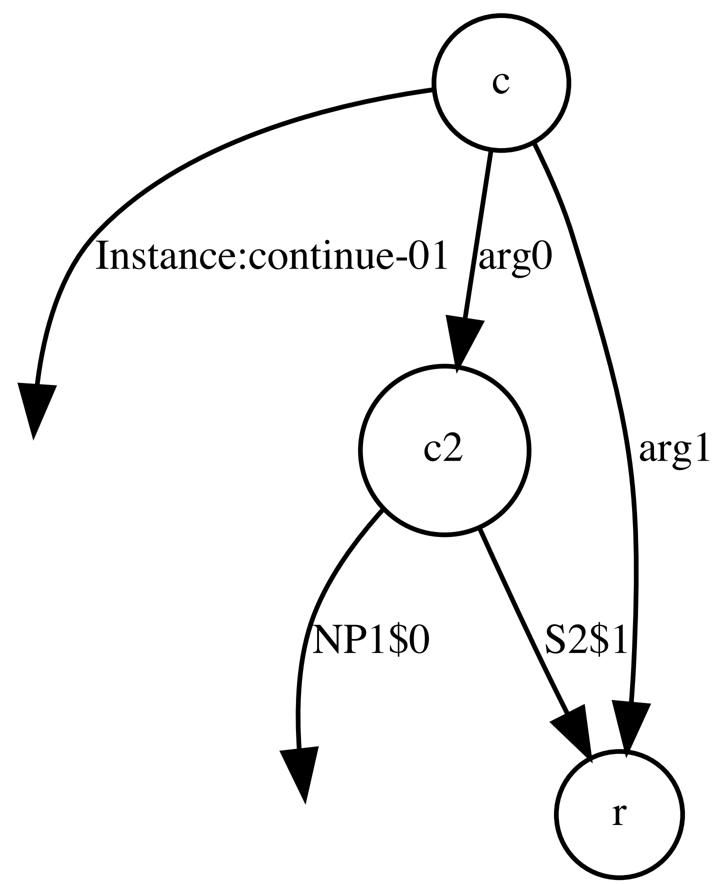
- Select 1st best Viterbi tree from each forest. Extract rules from these forests and keep trees as "bronze derivations"

31k rules



Extracted LSHRG Grammar Rule

S0 → NP1$_{[0]}$ VP-left* continue/t28 S2$_{[1]}$ S-right*



HRG Demo

- "Bolinas" package for Hyperedge Replacement Grammars.
  https://github.com/isi-nlp/bolinas

Acknowledgments

- Some slides from Nathan Schneider & Jeff Flanigan's AMR tutorial at NAACL 2015.