http://tiny.cc/amrtutorial

The Logic of AMR

Practical, Unified, Graph-Based Sentence Semantics for NLP

Nathan Schneider University of Edinburgh
Jeff Flanigan CMU
Tim O'Gorman CU-Boulder

The Logic of **AMR**

- Part I: **The AMR Formalism** (Tim & Nathan)

COFFEE BREAK

- Part II: **Algorithms and Applications** (Jeff)

Introduction to the Abstract Meaning Representation (AMR)

http://tiny.cc/amrtutorial

If you are here early, go to the **AMR Editor** and try to log in:

http://tiny.cc/amreditor

Why does AMR matter now?

- AMR is a semantic representation aimed at **large-scale human annotation** in order to **build a giant semantics bank.**

- We do a practical, replicable amount of abstraction (**limited canonicalization**).

- Capture many aspects of meaning in **a single simple data structure.**

Hasn't this been done before?

- Linguistics/CL have formalized semantics for a long time.
- A form of AMR has been around for a long time too (Langkilde and Knight 1998).
- It changed a lot since 1998 (add PropBank, etc.) and **we actually built a corpus of AMRs.**

Contemporary AMR

- **Banarescu et al. (2013)** laid out the fundamentals of the annotation scheme we'll describe today.

Roadmap for Part I

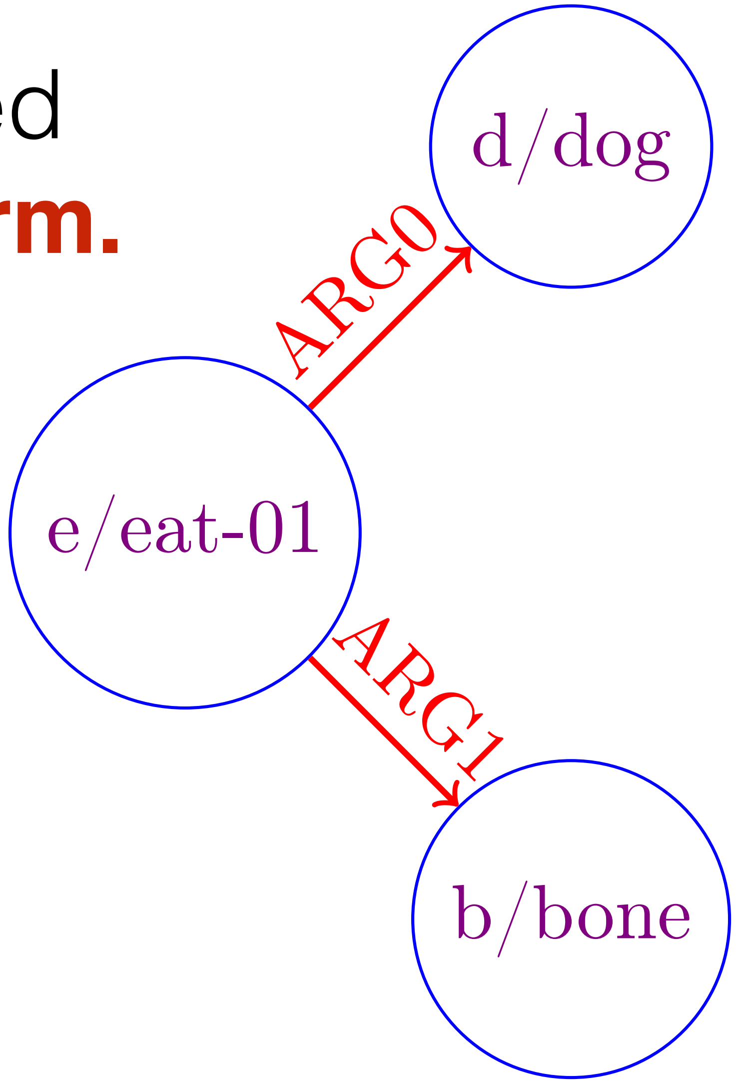
- Fundamentals of the AMR representation
- Hands-on practice I: Representing basic examples
- Advanced topics and linguistic phenomena
- Comparison to other representations
- Hands-on practice II: Doing real, complex text

PENMAN notation

- We use PENMAN notation (Bateman 1990).
- A way of representing a directed graph in a **simple, tree-like form.**

"*The dog is eating a bone*"

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

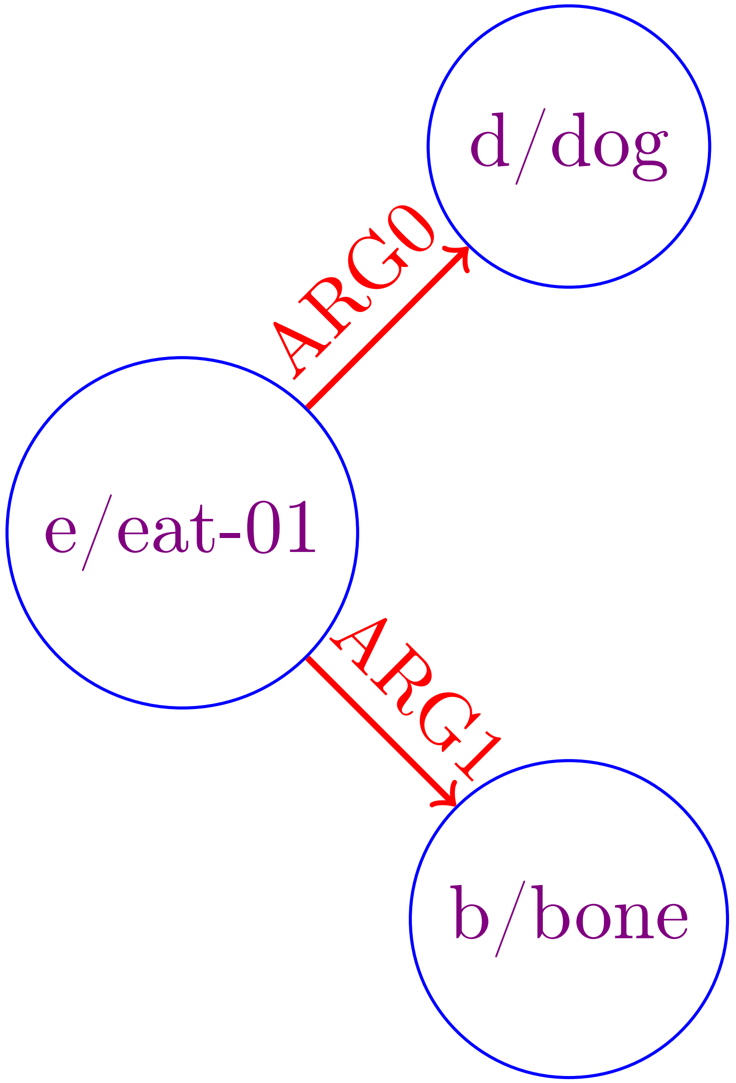


PENMAN notation

- The edges (ARG0 and ARG1) are **relations**
- Each node in the graph has a **variable**
- They are labeled with **concepts**
- **d** / **dog** means “**d** is an instance of **dog**”

“The dog is eating a bone”

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

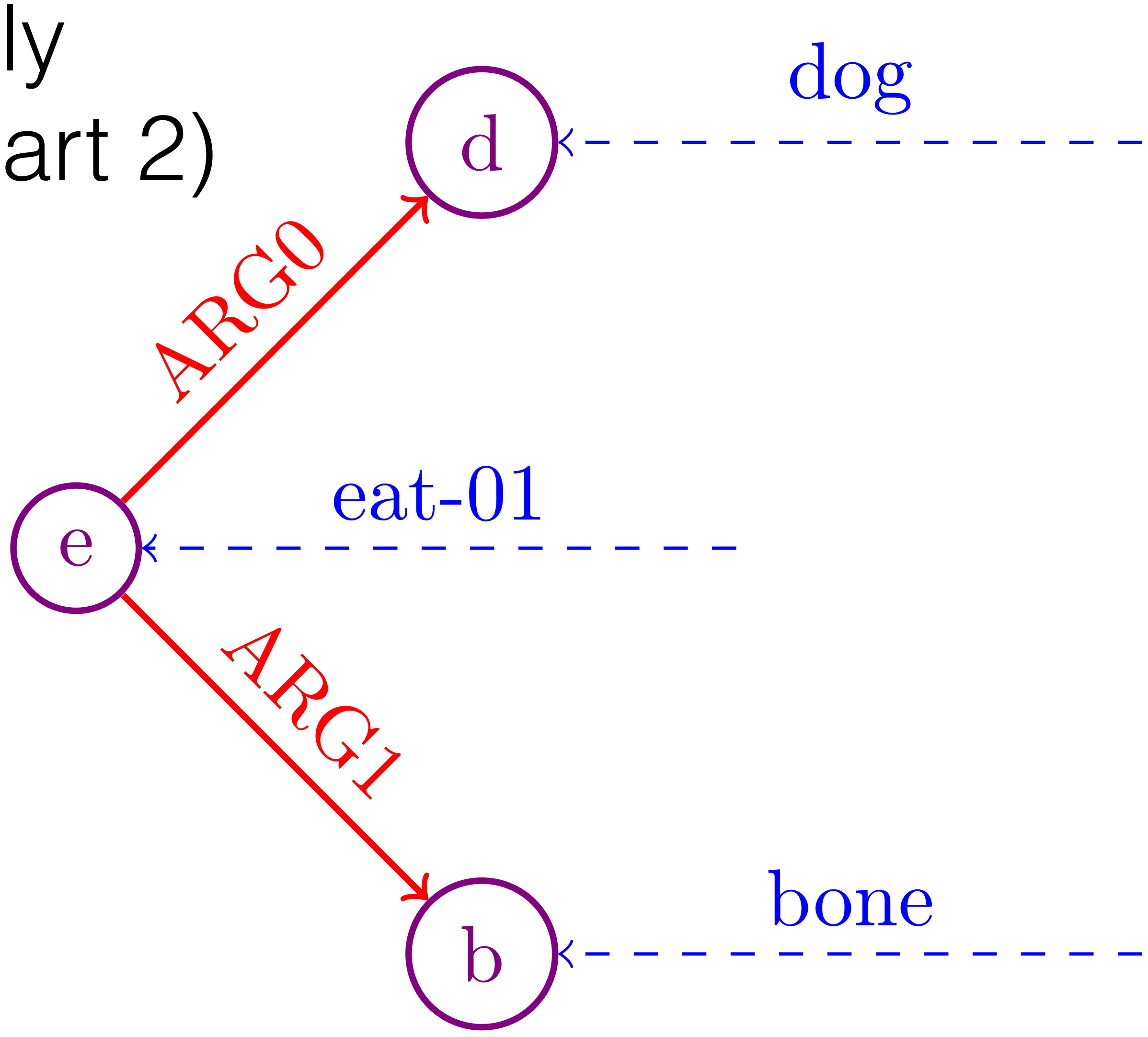


PENMAN notation

- **Concepts** are technically **edges** (this matters in Part 2)

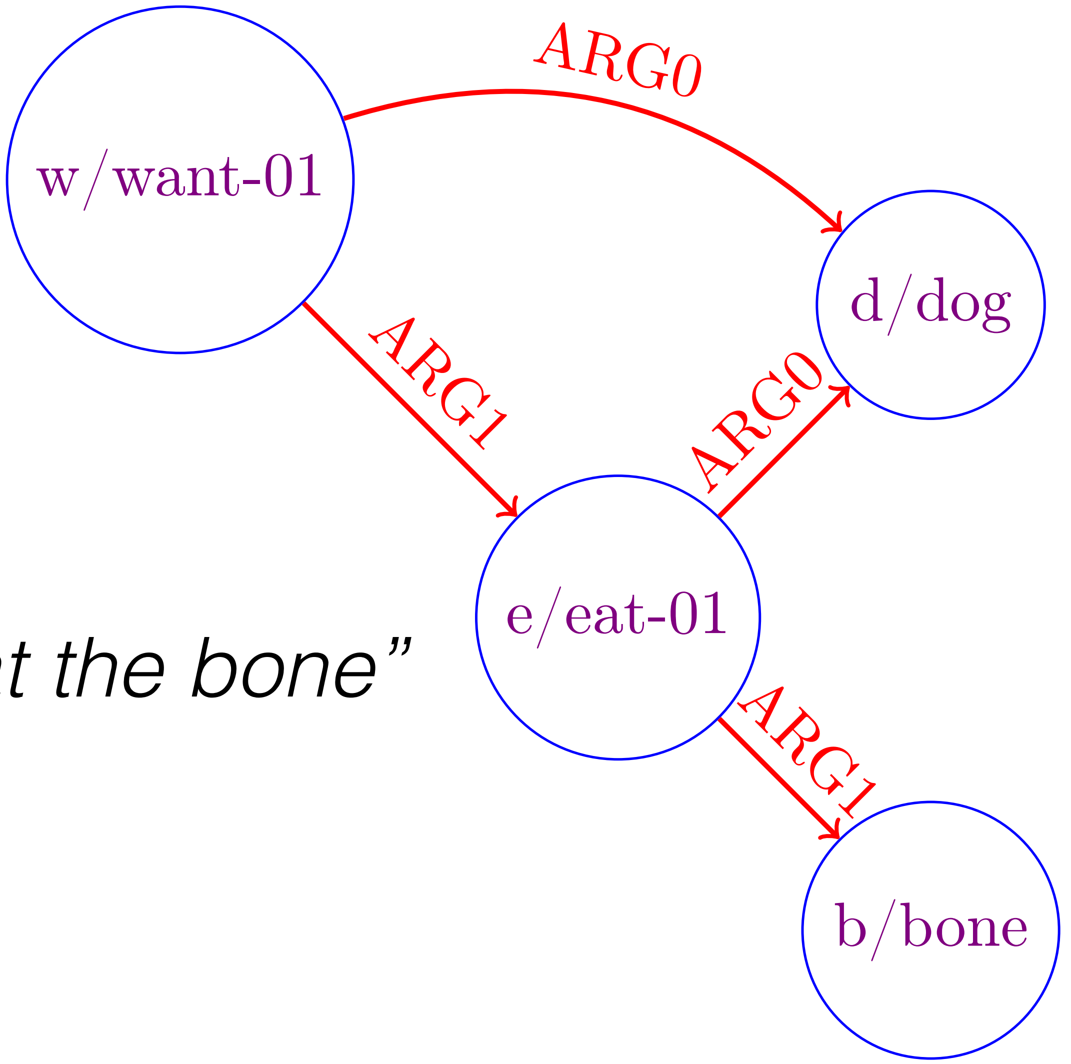
"The dog is eating a bone"

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



Reentrancy

- What if something is referenced multiple times?
- Notice how **dog** has two incoming roles now.
- To do this in PENMAN format, repeat the variable. We call this a **reentrancy**.



"The dog ***wants to*** *eat the bone"*

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

Reentrancy

- It **does not matter** where the concept label goes.

"*The dog wants to eat the bone*"

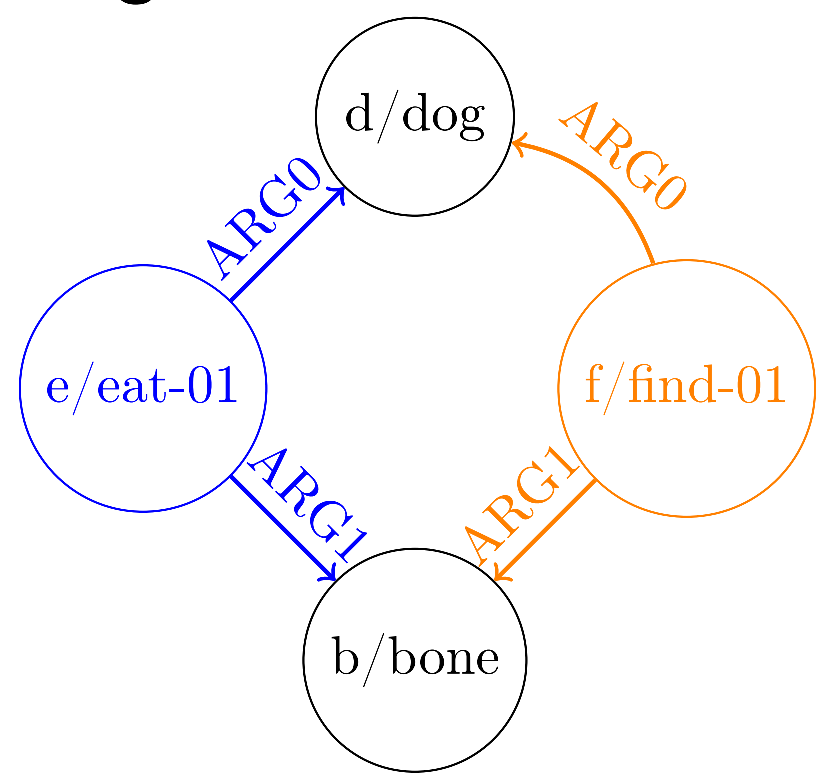
```
(want-01
    :ARG0 (d / dog)
    :ARG1 (e /eat-01
        :ARG0 d
        :ARG1 (b / bone)))
```

=

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

Inverse Relations and Focus

- What about “The dog ate **the bone that he found**”?



- How do we know **what goes on top**?
- How do we get these **into the AMR format**?

Inverse Relations and Focus

- We call "what goes on top" the **focus**.
- Conceptually, **the main assertion**.
- Linguistically, **often the head**.
  - For a sentence, **usually the main verb**.

Inverse Relations and Focus

The man at the hotel

FOCUS

The hotel the man is at

FOCUS

The dog ran

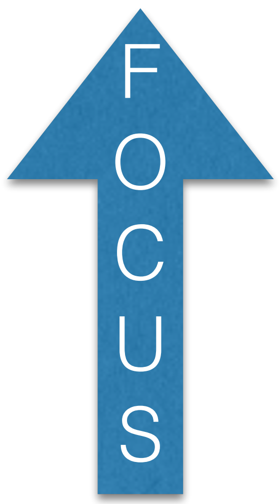
FOCUS

The dog that ran

FOCUS

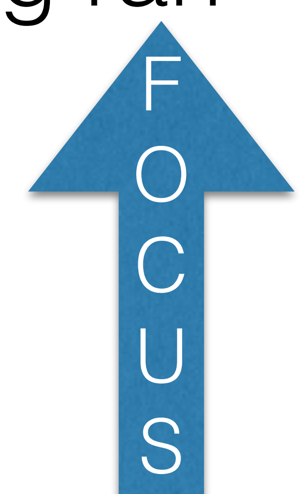
Inverse Relations and Focus

The man at the hotel



(m / man
    :location (h / hotel))

The dog ran



(r / ran-01
    :ARG0 (d / dog))

Inverse Relations and Focus

The hotel the man is at

FOCUS

(h / hotel
    :??? (m / man))

The dog that ran

FOCUS

(d / dog
    :???? (r / ran-01))

Inverse Relations and Focus

The hotel the man is at

FOCUS

(h / hotel
    :location-of (m / man))

The dog that ran

FOCUS

(d / dog
    :ARG0-of (r / ran-01))

Inverse Relations and Focus

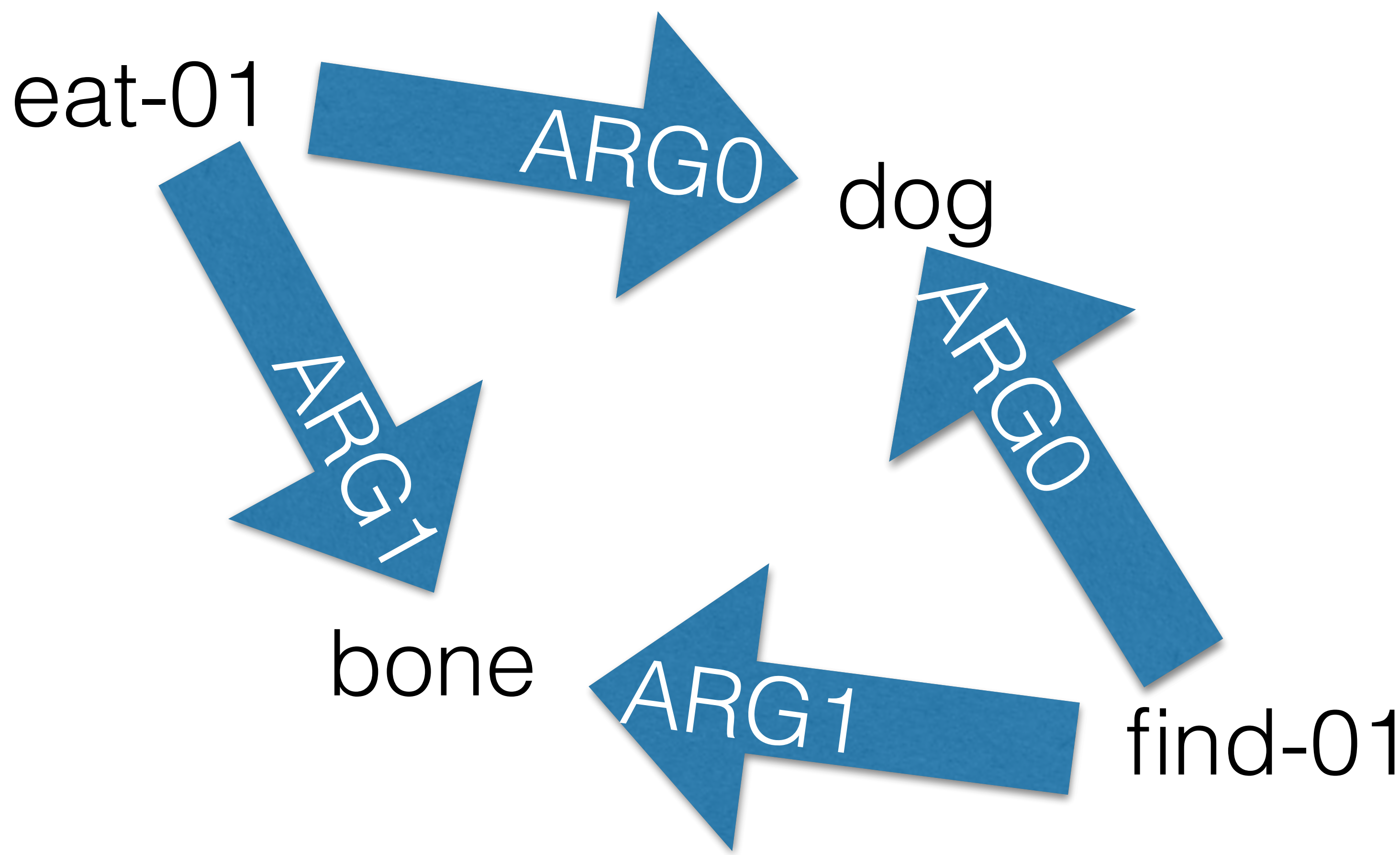
- This is a notational trick: **X ARG0-of Y = Y ARG0 X**

- Often used for relative clauses.

- *These are equivalent for SMATCH scoring purposes too.*

Reviewing the Format

- Imagine a graph for "***The dog ate the bone that he found***"

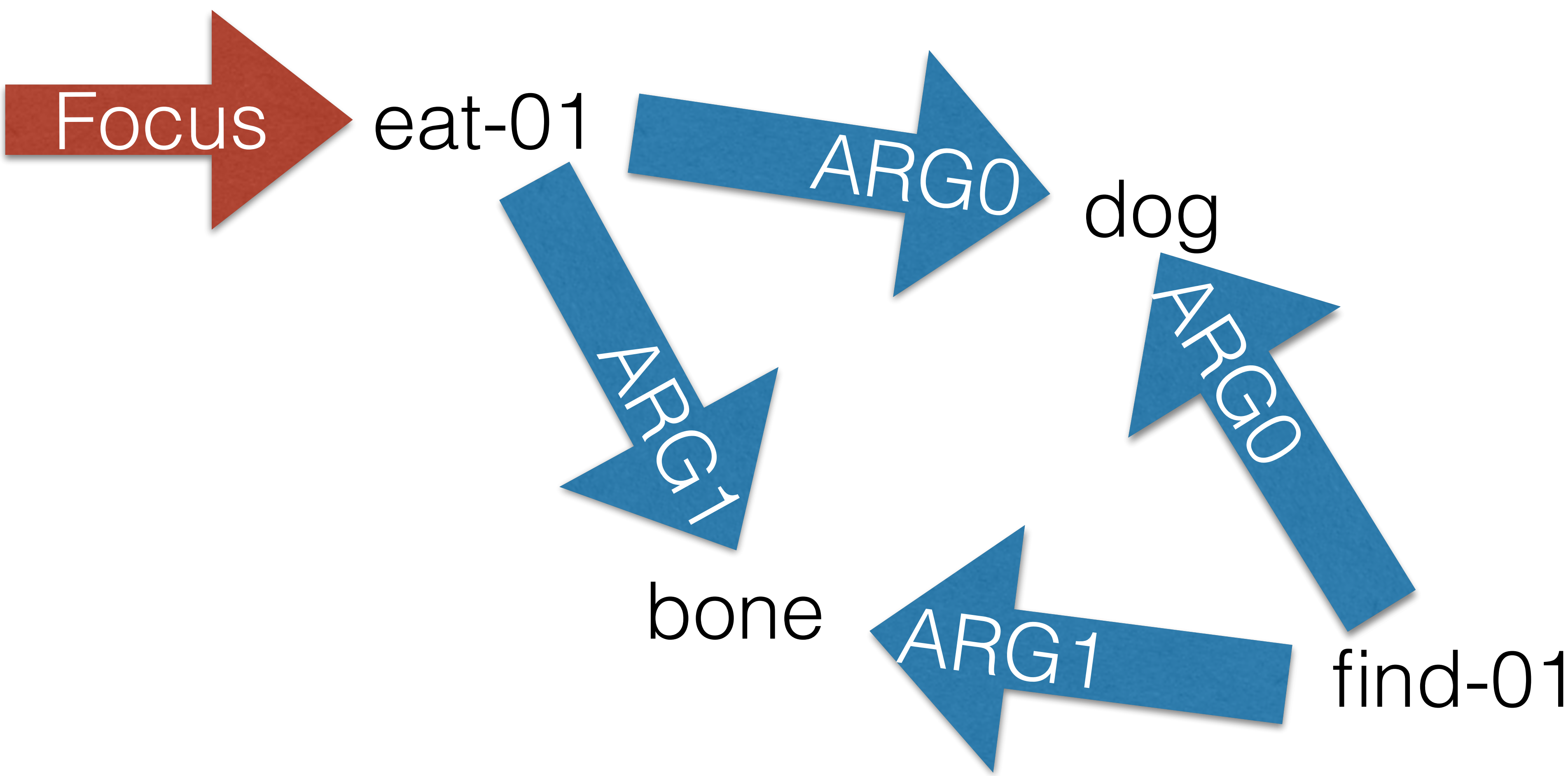


- "***The dog ate the bone that he found***"

Reviewing the Format

(e / eat-01 …)

- Find the focus

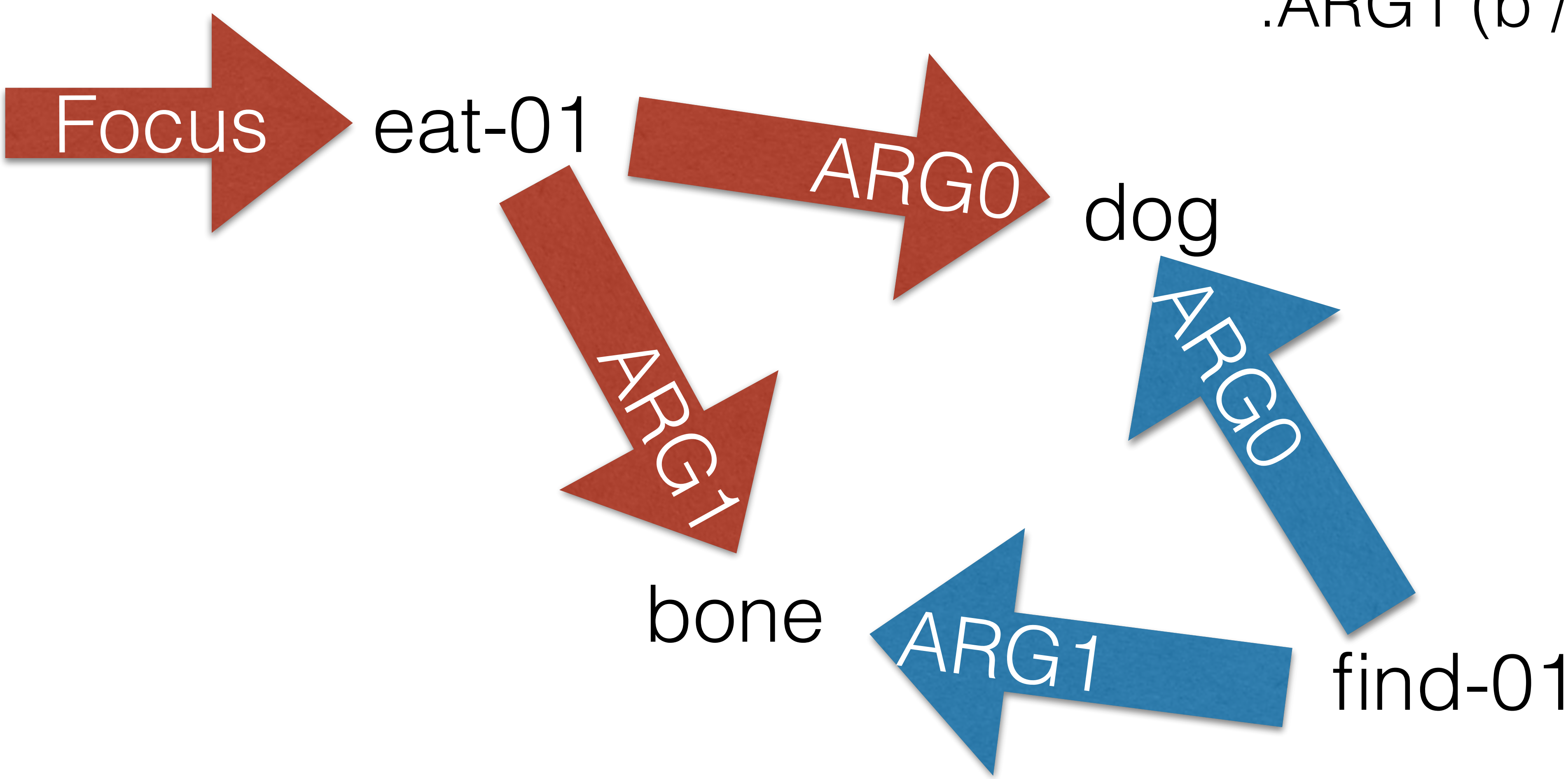


- "***The dog ate the bone that he found***"

Reviewing the Format

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

- Add entities

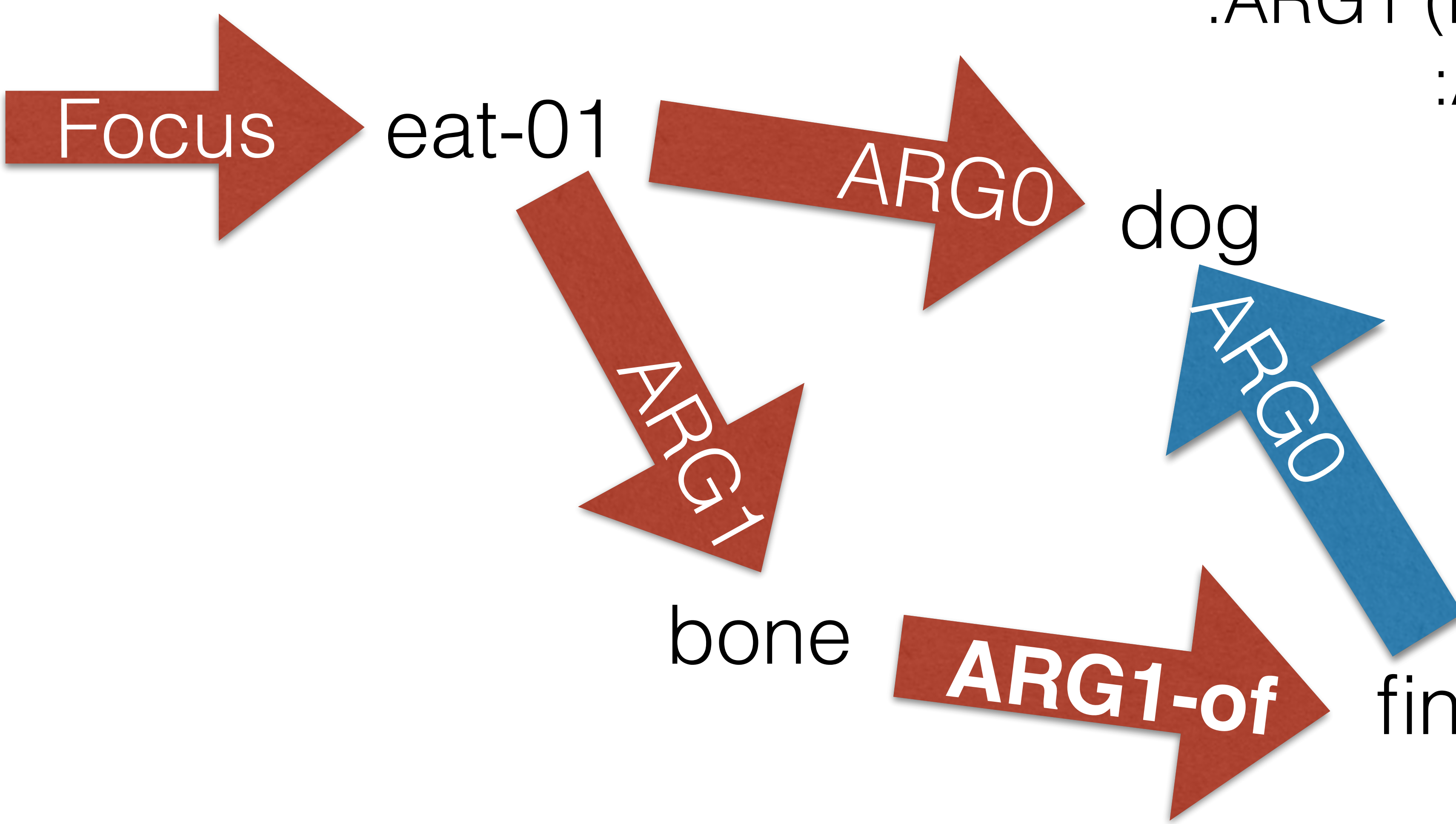


- "***The dog ate the bone that he found***"

Reviewing the Format

- Invert a relation if needed

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

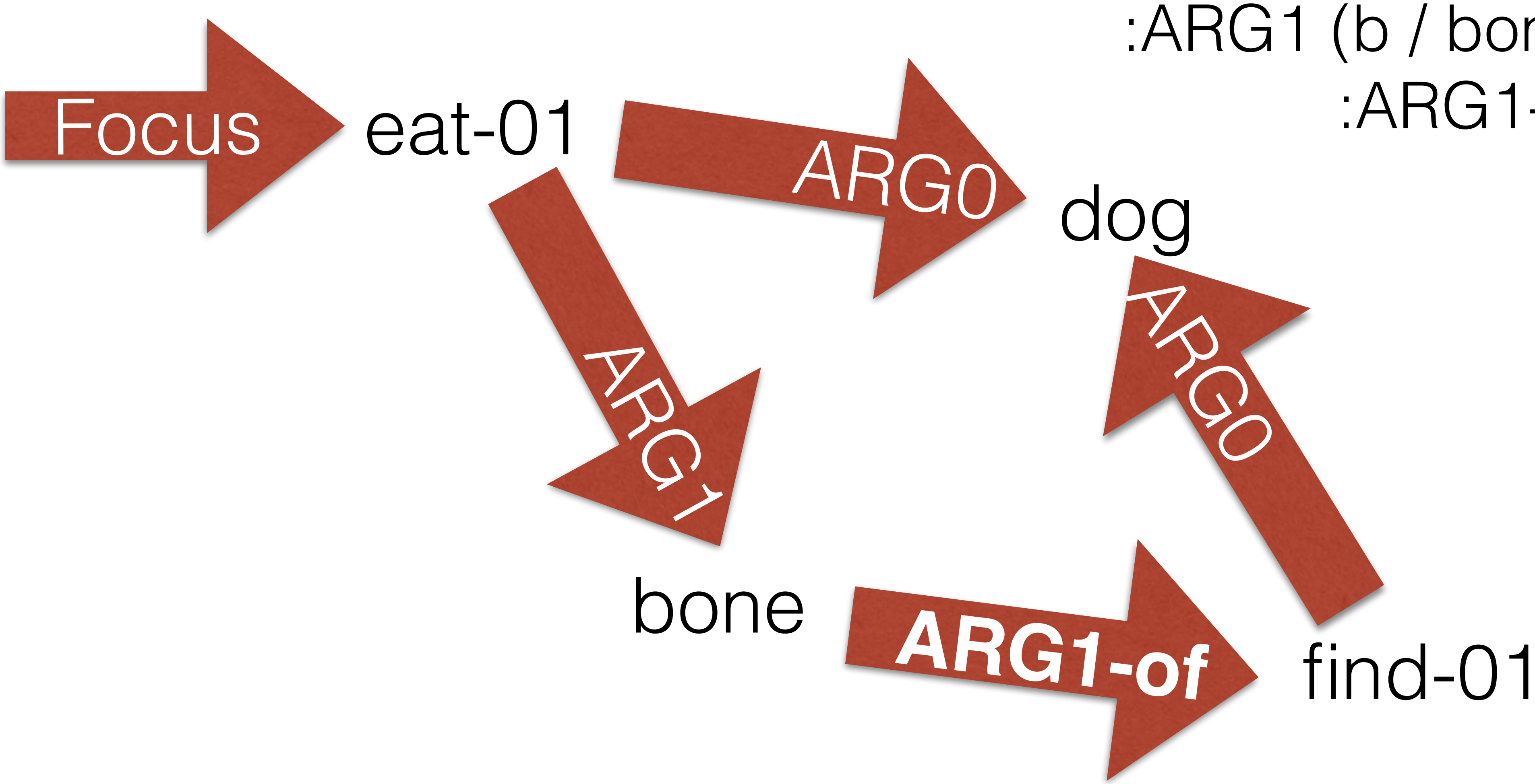


- "*The dog ate the bone that he found*"

Reviewing the Format

- Add reentrancies

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



- "***The dog ate the bone that he found***"

Constant

- Some relations, called **constants**, get no **variable**.
- The editor does this **automatically** for certain contexts.
- This happens for **negation**.

*"The dog **did not** eat the bone"*

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

Constant

- Some relations, called **constants**, get no **variable**.
- The editor does this **automatically** for certain contexts.
- This happens for **numbers**.

*"The dog ate **four** bones"*

```
(e /eat-01
   :ARG0 (d / dog)
   :ARG1 (b / bone :quant 4))
```

(to create a concept starting with a nonalphabetic character, type "!" before the concept)

Constant

- Some relations, called **constants**, get no **variable**.
- The editor does this **automatically** for certain contexts.
- This happens for **names**

*“**Fido** the dog”*

```
(d / dog
   :name (n / name :op1 "Fido"))
```

Concepts vs. Constants

- A **concept** is a type. For every concept node there will be ≥1 instance variable/node.
  - An instance can be mentioned multiple times.
  - Multiple instances of the same concept can be mentioned.
- **Constants** are singleton nodes: no variable, just a value. Specific non-core roles allow constant values.

- That's AMR notation! Let's review before discussing **how we annotate AMRs**.

variable **concept** **constant**

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

inverse relation **reentrancy**

PropBank Lexicon

- Predicates use the ***PropBank*** *inventory.*
- Each frame presents annotators with a list of senses.
- Each sense has its own definitions for its numbered (core) arguments

run-01 - "operate, proceed, operate or proceed"

- ARG0: operator
- ARG1: machine, operation, procedure
- ARG2: employer
- ARG3: coworker
- ARG4: instrumental

Aliases: run (v), run (n), running (n)
more

---

run-02 - "walk quickly, a course or contest, run/jog, run for office"

- ARG0: runner theme
- ARG1: course, race, distance location
- ARG2: opponent

Aliases: run (v), run (n), running (n)
more

---

run-03 - "cost"

- ARG1: commodity
- ARG2: price
- ARG3: buyer

Aliases: run (v), running (n)

PropBank Lexicon

- We generalize across **parts of speech** and **etymologically related words**:

My fear of snakes	fear-01
I am fearful of snakes	fear-01
I fear snakes	fear-01
I'm afraid of snakes	fear-01

- But we **don't** generalize over synonyms:

My fear of snakes	fear-01
I'm terrified of snakes	terrify-01
Snakes creep me out	creep_out-03

Stemming Concepts

- Non-predicates don’t have PropBank frames. They are simply stemmed.
- All concepts drop **plurality**, **articles**, and **tense**.

A cat	eating
The cat	eats
cats	ate
the cats	will eat
(c / cat)	(e / eat-01)

Why drop articles?

- All mentions of a term go to **the same variable**, including **pronouns** and **later nominal mentions**.

*I saw **a nice dog** and noticed **he** was eating a bone*

```
(d / dog
    :mod nice)
```

Is “d” indefinite or definite?

- We do capture **demonstratives**:

This house

```
(h / house
    :mod (t / this))
```

Stemming Concepts

- Pronouns that do not have a coreferent nominal mention are **made nominative** and **used as normal concepts.**

The man saved himself	**He** saved himself	He saved **me**
(s / save-01 :ARG0 (m / man) :ARG1 m)	(s / save-01 :ARG0 (**h / he**) :ARG1 h)	(s / save-01 :ARG0 (h / he) :ARG1 (**i / i**))

Why drop tense?

- English verbal tense **doesn't generalize well cross-linguistically**; not available for nominal predicates.

- Richer time representation might have required looking beyond a sentence.

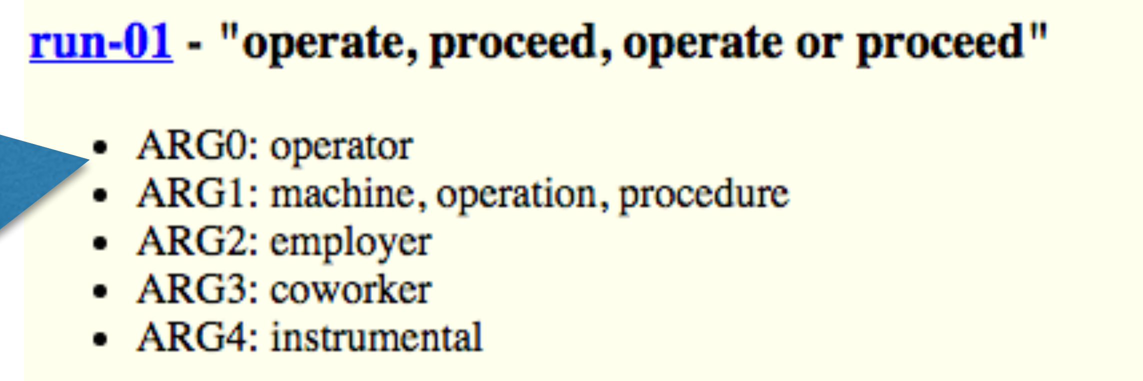
- Keep a simple representation.

The man described the mission as a disaster.
The man's description of the mission: disaster.
As the man described it, the mission was a disaster.
The man described the mission as disastrous.

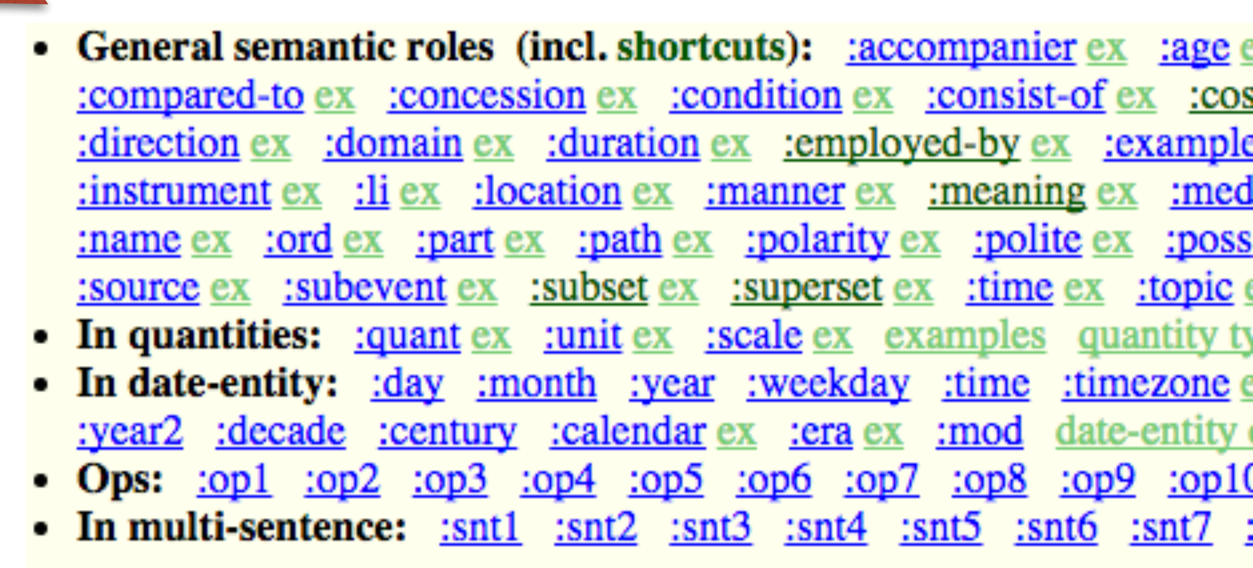
```
(d / describe-01
    :ARG0 (m / man)
    :ARG1 (m2 / mission)
    :ARG2 (d / disaster))
```

Non-core Role Inventory

- If a semantic role is not in the **core roles** for a roleset, AMR provides an inventory of **non-core roles**
- These express things like **:time, :manner, :part, :location, :frequency**
- Inventory on handout, or in editor (the **[roles]** button)


run-01 - **"operate, proceed, operate or proceed"**

- ARG0: operator
- ARG1: machine, operation, procedure
- ARG2: employer
- ARG3: coworker
- ARG4: instrumental


- **General semantic roles (incl. shortcuts):** :accompanier ex :age e :compared-to ex :concession ex :condition ex :consist-of ex :cos :direction ex :domain ex :duration ex :employed-by ex :example :instrument ex :li ex :location ex :manner ex :meaning ex :med :name ex :ord ex :part ex :path ex :polarity ex :polite ex :poss :source ex :subevent ex :subset ex :superset ex :time ex :topic e
- **In quantities:** :quant ex :unit ex :scale ex examples quantity ty
- **In date-entity:** :day :month :year :weekday :time :timezone e :year2 :decade :century :calendar ex :era ex :mod date-entity
- **Ops:** :op1 :op2 :op3 :op4 :op5 :op6 :op7 :op8 :op9 :op10
- **In multi-sentence:** :snt1 :snt2 :snt3 :snt4 :snt5 :snt6 :snt7 :

Non-core Role Inventory

- We use **:mod** for attribution, and **:domain** is the inverse of mod (**:domain** = **:mod-of**)

The yummy food
There is yummy food

```
(f / food
   :mod (y / yummy))
```

The yumminess of the food
The food is yummy

```
(y / yummy
   :domain (f / food))
```

seeing the yummy food
seeing the food that is yummy

```
(s / see-01
   :ARG1 (f / food
      :mod (y / yummy)))
```

seeing ***that*** *the food is yummy*

```
(s / see-01
   :ARG1 (y / yummy
      :domain (f / food)))
```

Non-core Role Inventory

- This is also used for attribute/predicative demonstratives and nominals

This house

(h / house
  :mod (t / this))

A monster truck

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

the truck is a monster

(m / monster
  :domain (t / truck))

Non-core Roles: **:op#**

- Some relations need to have an ordered list of arguments, but **don't have specific meanings** for each entry.
- We use **:op1**, **:op2**, **:op3**, … for these

:op# for coordination

- We use this for coordination:
- *Apples and bananas*

```
(a / and
    :op1 (a2 / apple)
    :op2 (b / banana))
```

:op# for names

- *Barack Obama*

```
(p / person
    :name (n / name
        :op1 "Barack"
        :op2 "Obama"))
```

- *Obama*

```
(p / person
    :name (n / name
        :op1 "Obama"))
```

Named Entities

- *Barack Obama*

```
(p / person
    :name (n / name
        :op1 "Barack"
        :op2 "Obama"))
```

- Entities with names get special treatment!
- We assign a **named entity type** from our ontology.
- 70+ categories like ***person, criminal-organization, newspaper, city, food-dish, conference***
- See your handout, or the **[NE types]** button in the editor

Named Entities

- *Barack Obama*

```
(p / person
    :name (n / name
        :op1 "Barack"
        :op2 "Obama"))
```

- Entities with names get special treatment!

- Each gets a **:name** relation to a **name node**

- That node gets :op# relations to the strings of their name ***as used in the sentence***.

Named Entities

- If there is a more specific descriptor present in the sentence, we use that **instead of the NE inventory.**

- *a Kleenex*

```
(p / product
      :name (n / name
            :op1 "Kleenex"))
```

- *a Kleenex tissue*

```
(t / tissue
      :name (n / name
            :op1 "Kleenex"))
```

Wikification

- In a second pass of annotation, we add **:wiki** relations.
- *Barack Obama*

```
(p / person
      :name (n / name
            :op1 "Barack"
            :op2 "Obama")
      :wiki Barack_Obama)
```

- http://en.wikipedia.org/wiki/**Barack_Obama**

Measurable Entities

- We also have special entity types we use for **normalizable entities**.

"Tuesday the 19th"

(**d / date-entity**
:weekday (t / tuesday)
:day 19)

"five bucks"

(**m / monetary-quantity**
:unit dollar
:quant 5)

Measurable Entities

- We also have special entity types we use for **normalizable entities**.

"$3 / gallon"

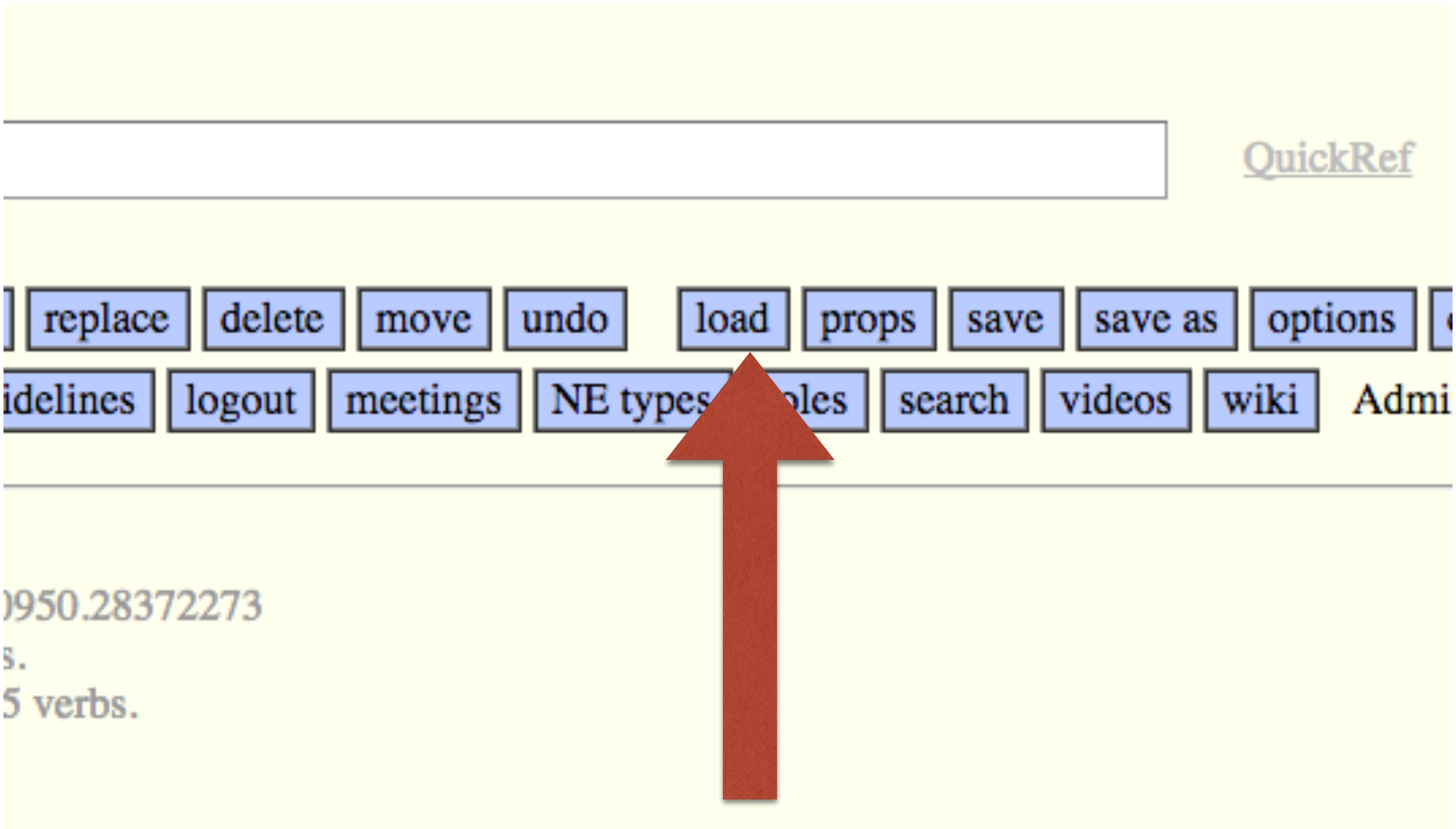
```
(r / rate-entity-91
    :ARG1 (m / monetary-quantity
        :unit dollar
        :quant 3)
    :ARG2 (v / volume-quantity
        :unit gallon
        :quant 1))
```

Hands-on Annotation!

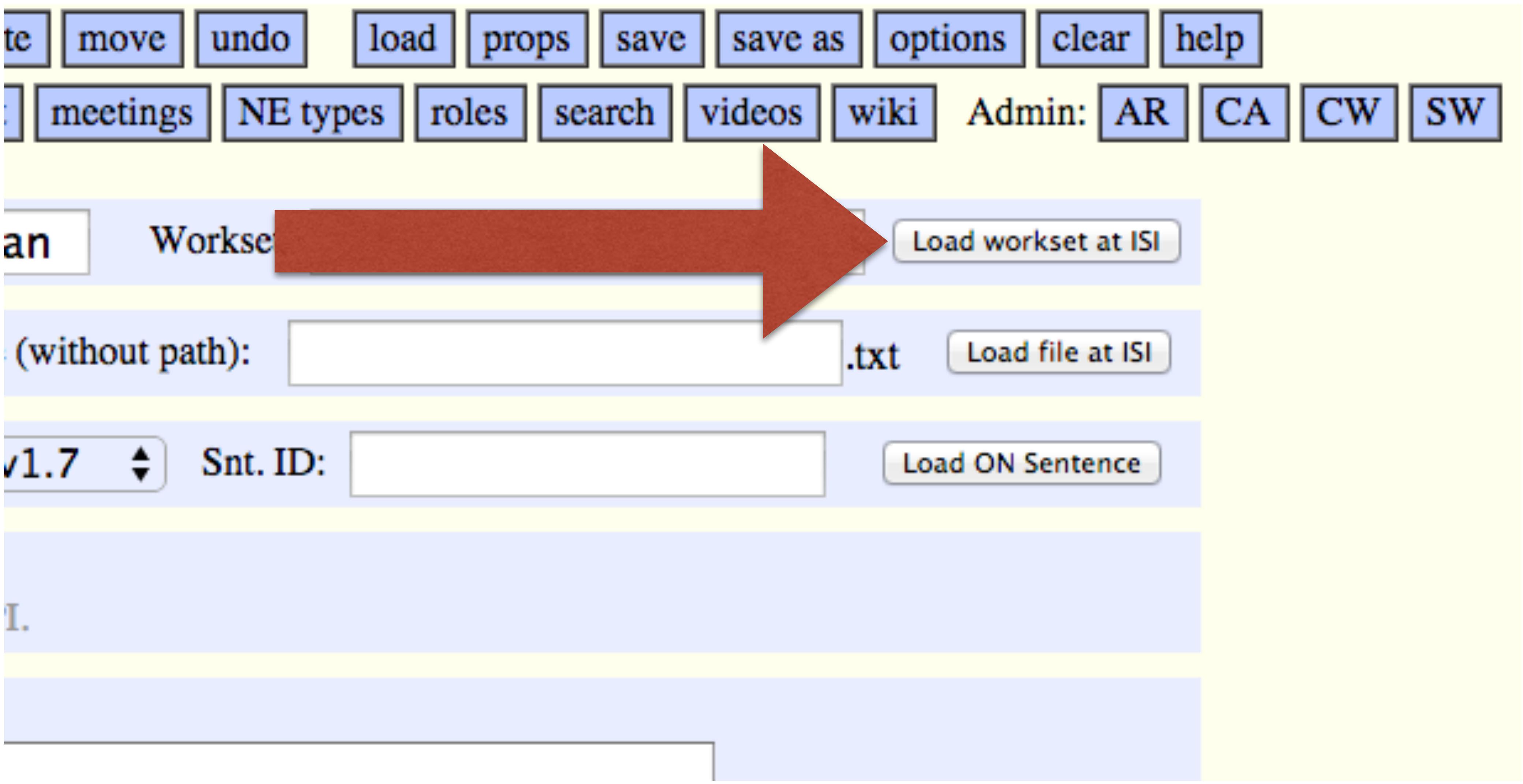
Go to the **AMR Editor**:

http://tiny.cc/amreditor

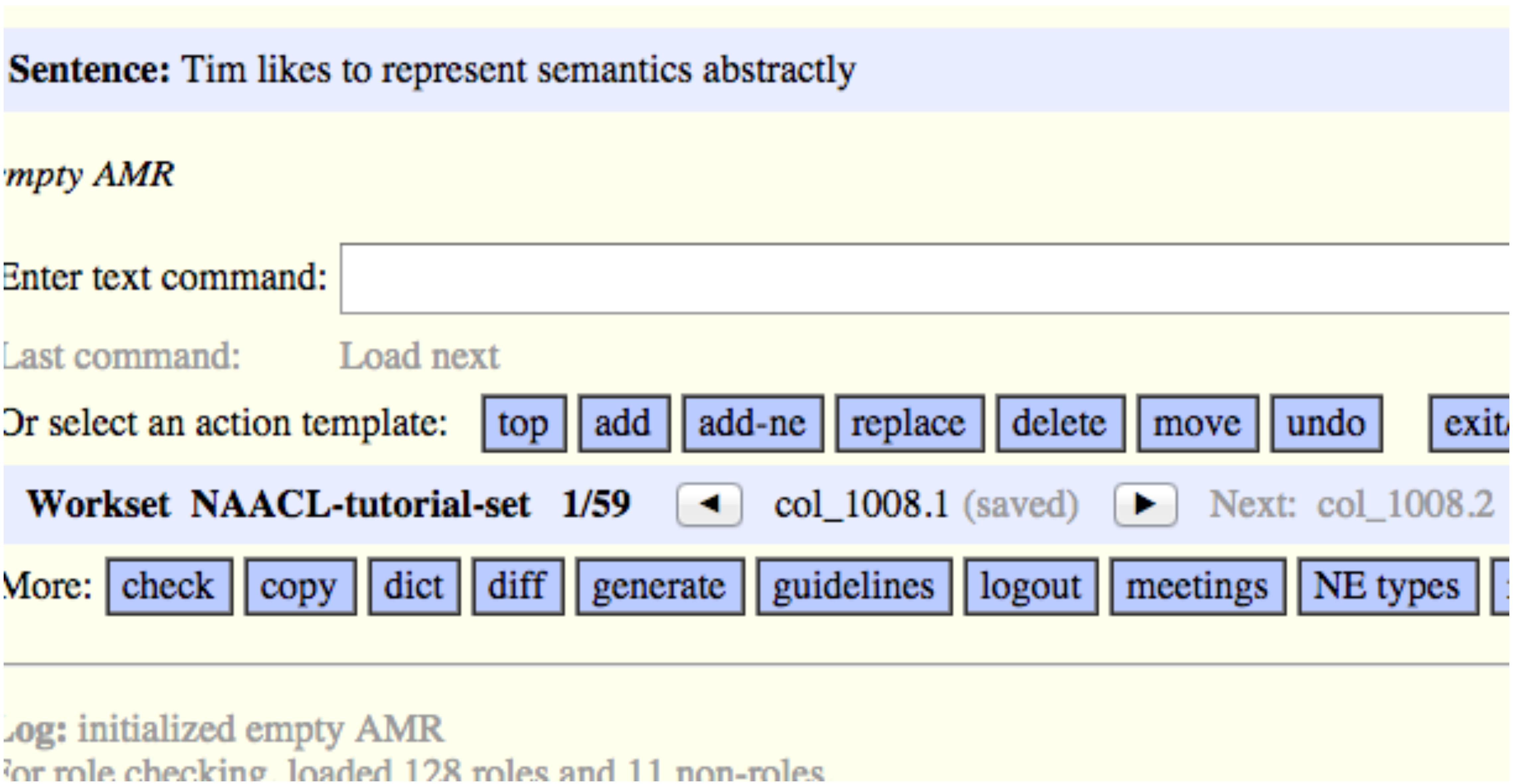
Load the Tutorial Sentences



Select “NAACL Tutorial”

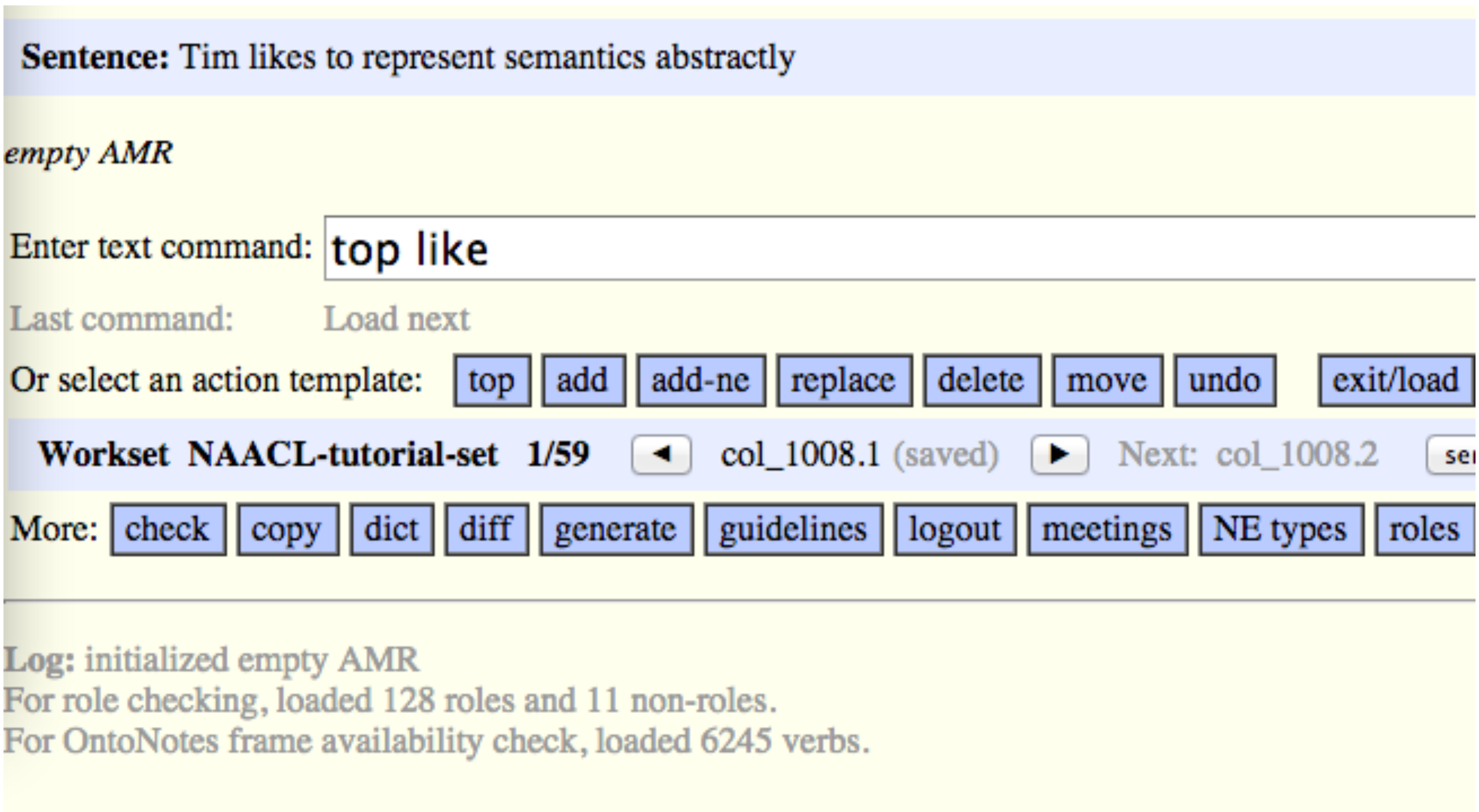


It should look like this



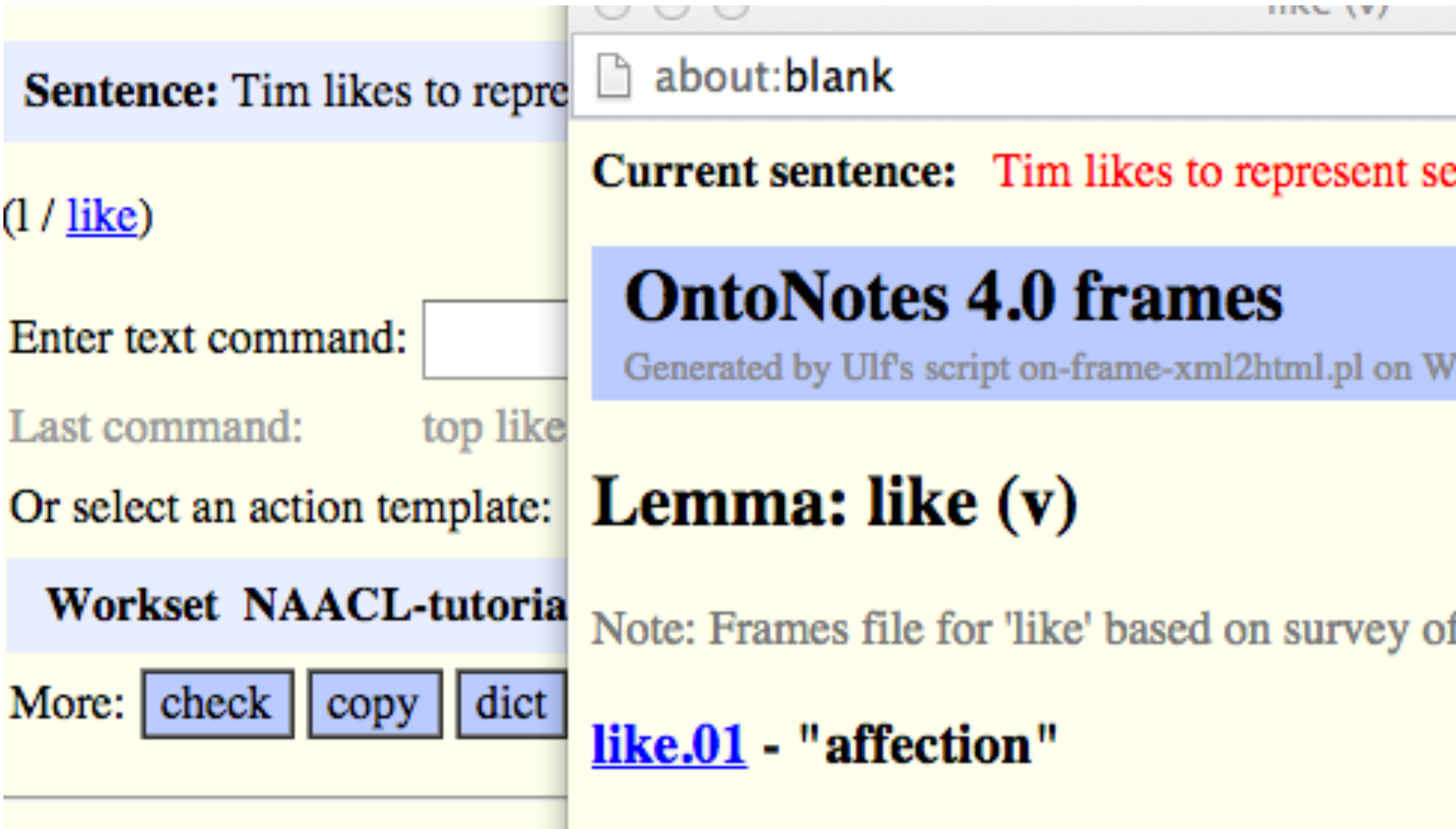
Commands

Use “top <concept>” to make a top node



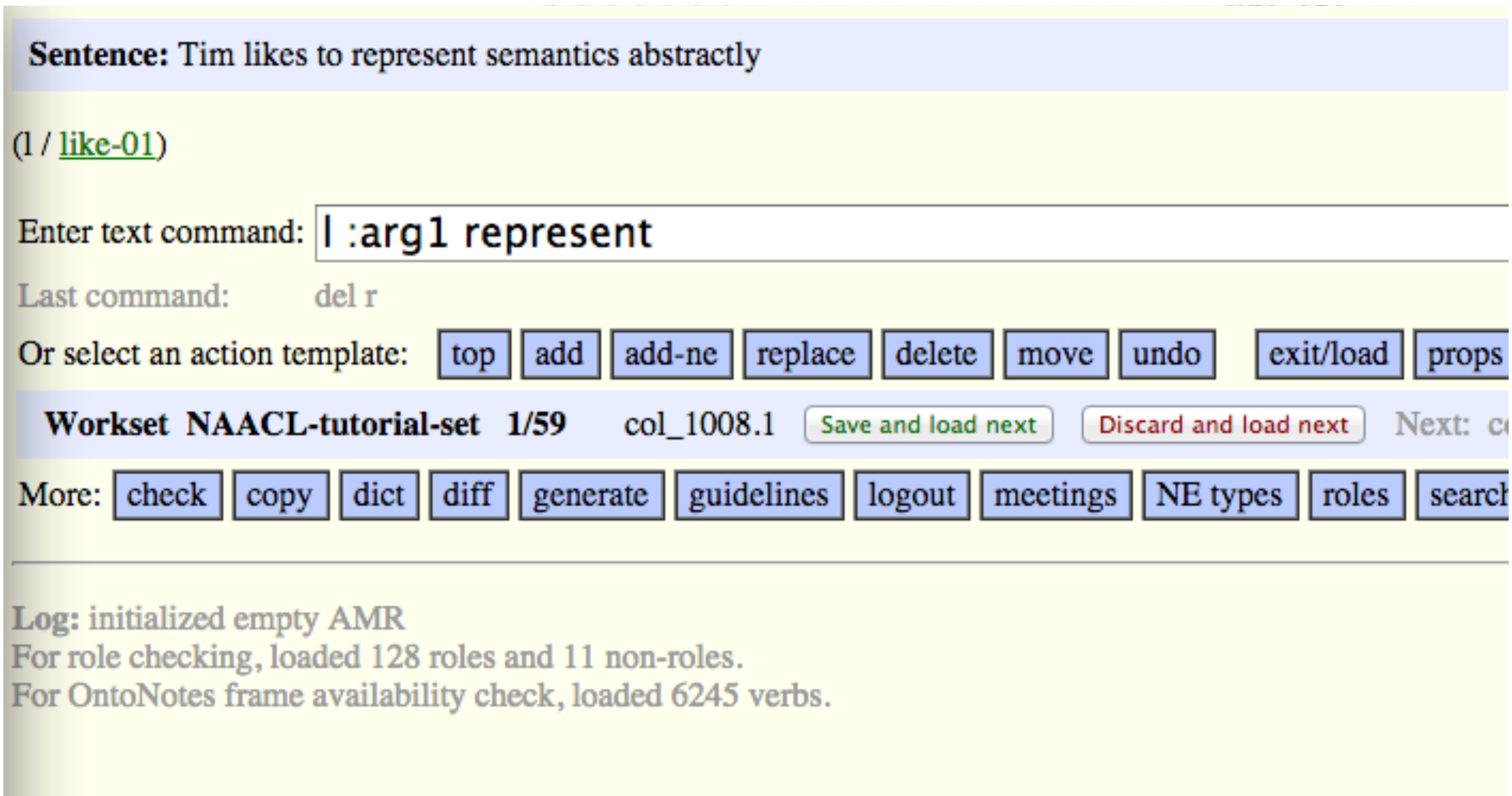
Commands

Click on “like” to select the right sense



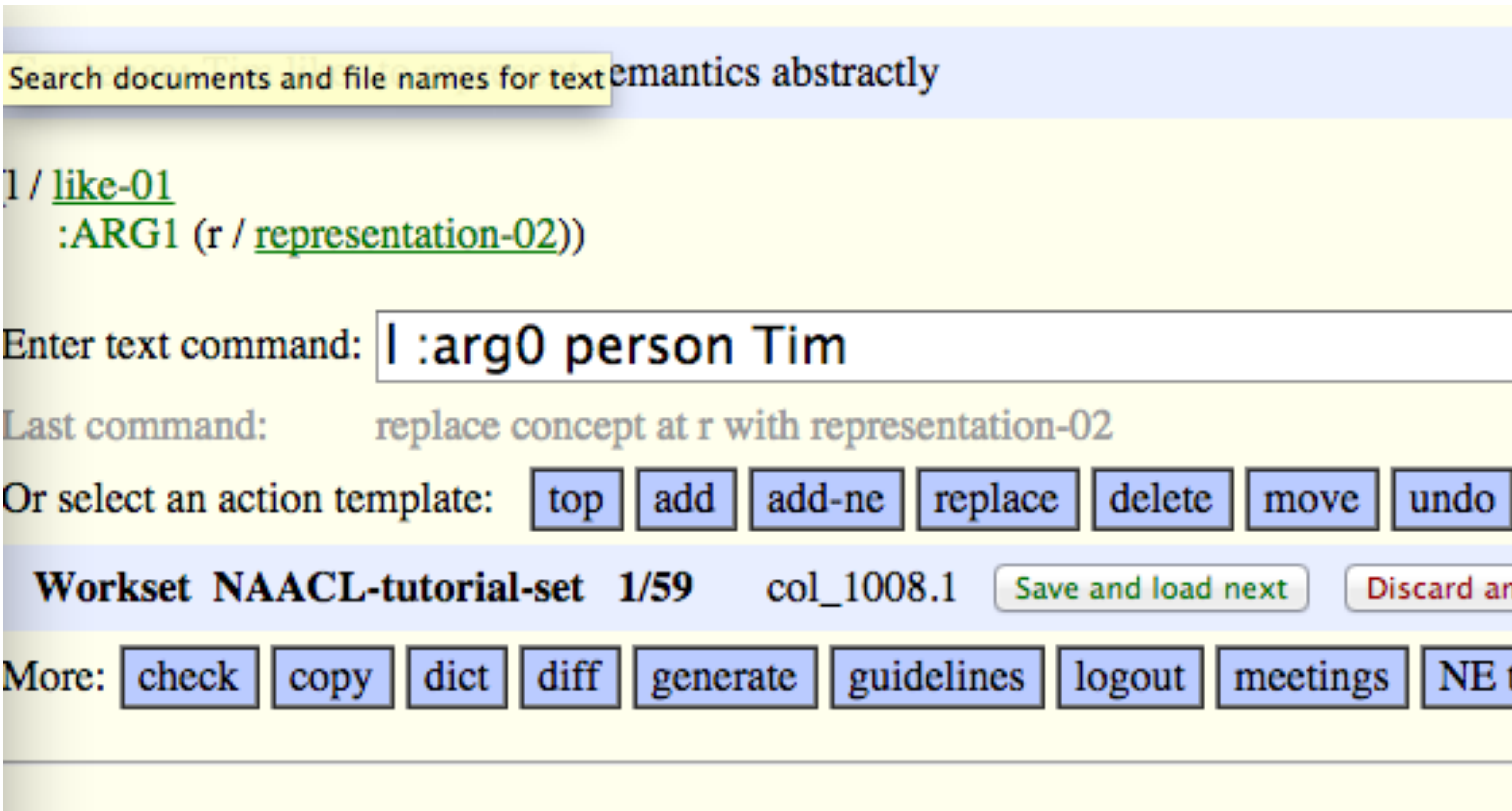
Commands

New relation: <variable> :<role> <concept>



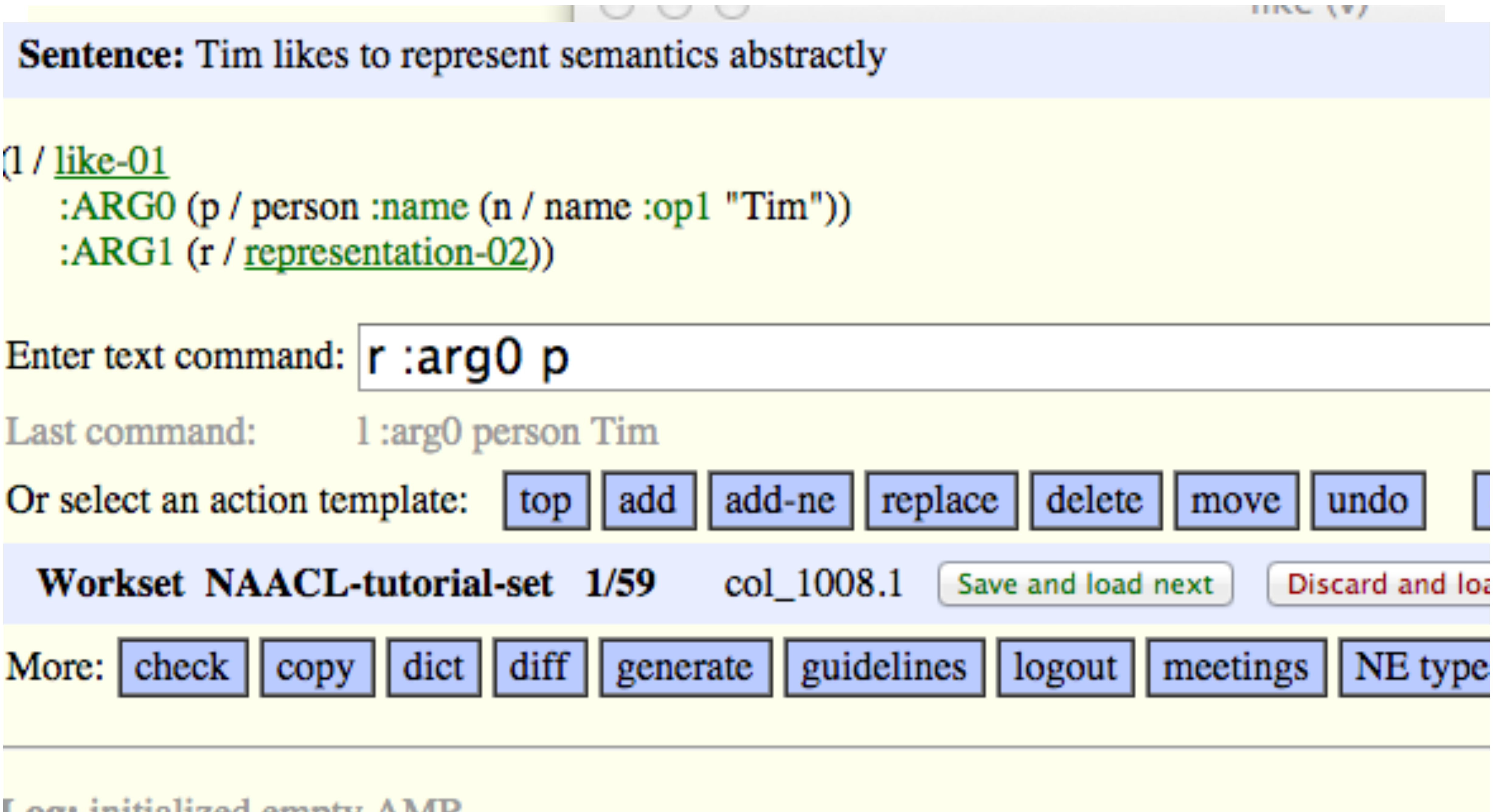
Commands

Anything after the third element is made into a name



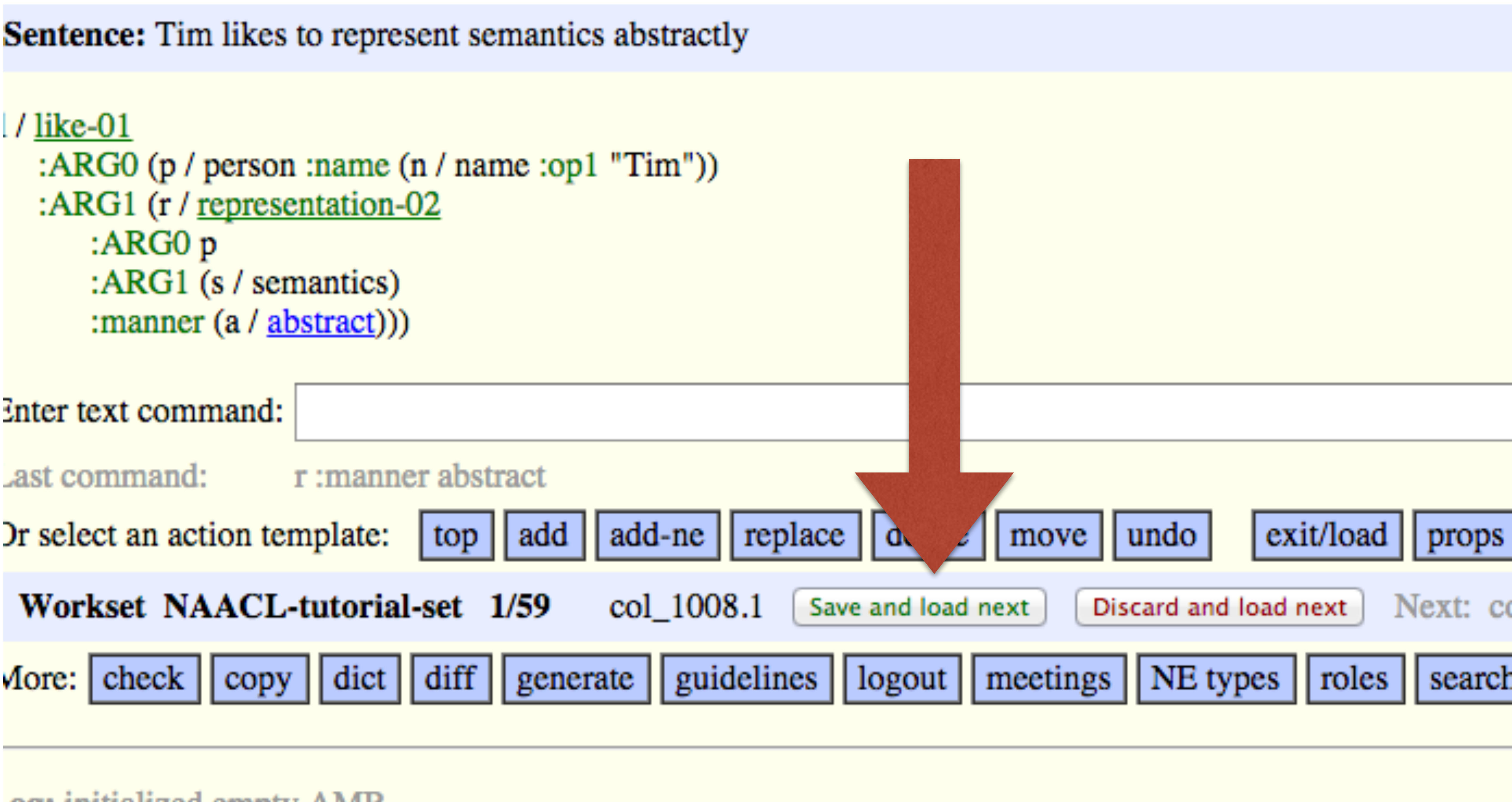
Commands

Make reentrancies with <variable> :<role> <variable>



Commands

When you are done, use “Save and Load Next”



Try the next sentence!

We will walk through it momentarily

AMR Annotation:
Special Topics

http://tiny.cc/amrtutorial

Congratulations!

You now know how to AMR **simple sentences**.

BUT: English is a wily opponent.

- Copulas, light verbs
- Prepositions
- Derivational morphology
- Relational nouns
- Coordination & clausal connectives
- Modality
- Non-declarative sentences
- Questions
- Comparisons
- Quantification
- Subsets
- Reification

Light semantics

- We try to eliminate purely grammatical words. E.g.:
  - **copulas:** *I **am** happy*

    (h / happy :domain (i / i))
  - **light verbs:** *I’m **taking** a bath*

    (b / bathe-01 :ARG0 (i / i))

Heavy prepositions

- Most prepositions mark a (core or non-core) role. Some add crucial additional information meriting a concept:

  - ***at*** *the school*: :location (s / school)
  - ***next to*** *the school*:
    :location (n / **next-to** :op1 (s / school))
  - ***between*** *the school and the house*:
    :location (b / **between** :op1 (s / school)
    :op2 (h / house))

  - ***at the time of*** *the war*: :time (w / war)
  - ***after*** *the war*: :time (a / **after** :op1 (w / war))

Typical uses of inverse roles

- **Relative clauses:** *someone **who** sifts thistles*
- **Derivational morphology**
  - **Participles:** *thistle-sift**ing** person*
  - **Nominalizations:** *thistle sift**er***

Concept may be implicit

- (p / person
  :ARG0-of (s / sift-01
  :ARG1 (t / thistle)))

Compositionality criterion

- We only “decompose” **derivational morphology** if a relative clause paraphrase is possible:
  - teach**er** = person who teaches

    (p / person :ARG0-of (t / teach-01))
  - profess**or** ≠ person who professes

    (p / professor)

Compositionality criterion

- Often core roles are available for modifiers:
  - math teach**er** / teach**er** of math
    = person who teaches math

    (p / person :ARG0-of (t / teach-01
    :ARG1 (m / math)))
  - math profess**or** ≠ person who professes math

    (p / professor :mod (m / math))

Compositionality criterion

- Sometimes it is difficult to draw a line, but we do our best:
    - opin**ion** = thing that is opined

      (t / thing :ARG1-of (o / opine-01))

    - profess**ion** ≠ thing that is professed

      (p / profession)

Hallucinating relations

- Sometimes we have to "hallucinate" a **relationship that the grammar underspecifies**.
  - e.g., possessives and noun-noun compounds can express many different kinds of relations

Relational nouns

- Special predicates for individual–group and individual–individual relations:

He is a pilot for TWA
He is a TWA pilot

```
(h / have-org-role-91
   :ARG0 (h2 / he)
   :ARG1 (c / company
     :name (n / name :op1 "TWA"))
   :ARG2 (p / pilot))
```

I am your father

```
(h / have-rel-role-91
   :ARG0 (i / i)
   :ARG1 (y / you)
   :ARG2 (f / father))
```

Coordination & Clausal Connectives

Example connectives	**AMR treatment**
and	`and`
or	`or`
but	`contrast-01`
because; due to; on account of	`:cause`
(in order) to; so (that)	`:purpose`
if	`:condition`
unless	`:condition (… :polarity -)`
although; despite	`:concession`

Coordination

- The most common patterns:

X, *Y*, and *Z*

```
(a / and
    :op1 X
    :op2 Y
    :op3 Z)
```

X, *Y*, or *Z*

```
(o / or
    :op1 X
    :op2 Y
    :op3 Z)
```

X but *Y*

```
(c / contrast-01
    :ARG1 X
    :ARG2 Y)
```

- “and”, “or” take 2 or more conjuncts in sequence as `:op#`

Coordination

Coordination

- Rachael Ray finds inspiration in cooking her family and her dog.

```
(i / inspire-01
   :ARG0 (c / cook-01








                                                  )
   :ARG1 (p / person :name (n / name :op1 "Rachael" :op2 "Ray")))
```

Coordination

- Rachael Ray finds inspiration in cooking her family and her dog.

```
(i / inspire-01
   :ARG0 (c / cook-01
            :ARG0 p
            :ARG1 (a / and
                     :op1 (f / family


                                                                   )
                     :op2 (d / dog        )))
   :ARG1 (p / person :name (n / name :op1 "Rachael" :op2 "Ray")))
```

Coordination

- Rachael Ray finds inspiration in cooking her family and her dog.

```
(i / inspire-01
   :ARG0 (c / cook-01
            :ARG0 p
            :ARG1 (a / and
                     :op1 (f / family


                                                                    )
                     :op2 (d / dog :poss p)))
   :ARG1 (p / person :name (n / name :op1 "Rachael" :op2 "Ray")))
```

Coordination

- Rachael Ray finds inspiration in cooking her family and her dog.

```
(i / inspire-01
   :ARG0 (c / cook-01
            :ARG0 p
            :ARG1 (a / and
                     :op1 (f / family
                              :ARG1-of (h / have-org-role-91
                                          :ARG0 p
                                          :ARG2 (m / member)))
                     :op2 (d / dog :poss p)))
   :ARG1 (p / person :name (n / name :op1 "Rachael" :op2 "Ray")))
```

X's family = family of which X is a member

Coordination

- Rachael Ray finds inspiration in cooking, her family, and her dog.

```
(i / inspire-01
   :ARG0 (a / and
            :op1 (c / cook-01
                    :ARG0 p)
            :op2 (f / family
                   :ARG1-of (h / have-org-role-91
                                :ARG0 p
                                :ARG2 (m / member)))
            :op3 (d / dog :poss p))
   :ARG1 (p / person :name (n / name :op1 "Rachael" :op2 "Ray")))
```

Coordination: shared core args

- We invited and then disinvited the students.

```
(a / and
   :op1 (i / invite-01
           :ARG0 (w / we)
           :ARG1 (s / student))
   :op2 (d / disinvite-01
           :ARG0 w
           :ARG1 s
           :time (t / then)))
```

Coordination: shared non-core args

- Yesterday we invited and then disinvited the students.

```
(a / and
   :op1 (i / invite-01
            :ARG0 (w / we)
            :ARG1 (s / student))
   :op2 (d / disinvite-01
            :ARG0 w
            :ARG1 s
            :time (t / then))
   :time (y / yesterday))
```

Coordination: copied predicates

- We invited the students and then the professors.

```
(a / and
   :op1 (i / invite-01
           :ARG0 (w / we)
           :ARG1 (s / student))
   :op2 (i2 / invite-01
           :ARG0 w
           :ARG1 (p / professor)
           :time (t / then)))
```

Modal Concepts

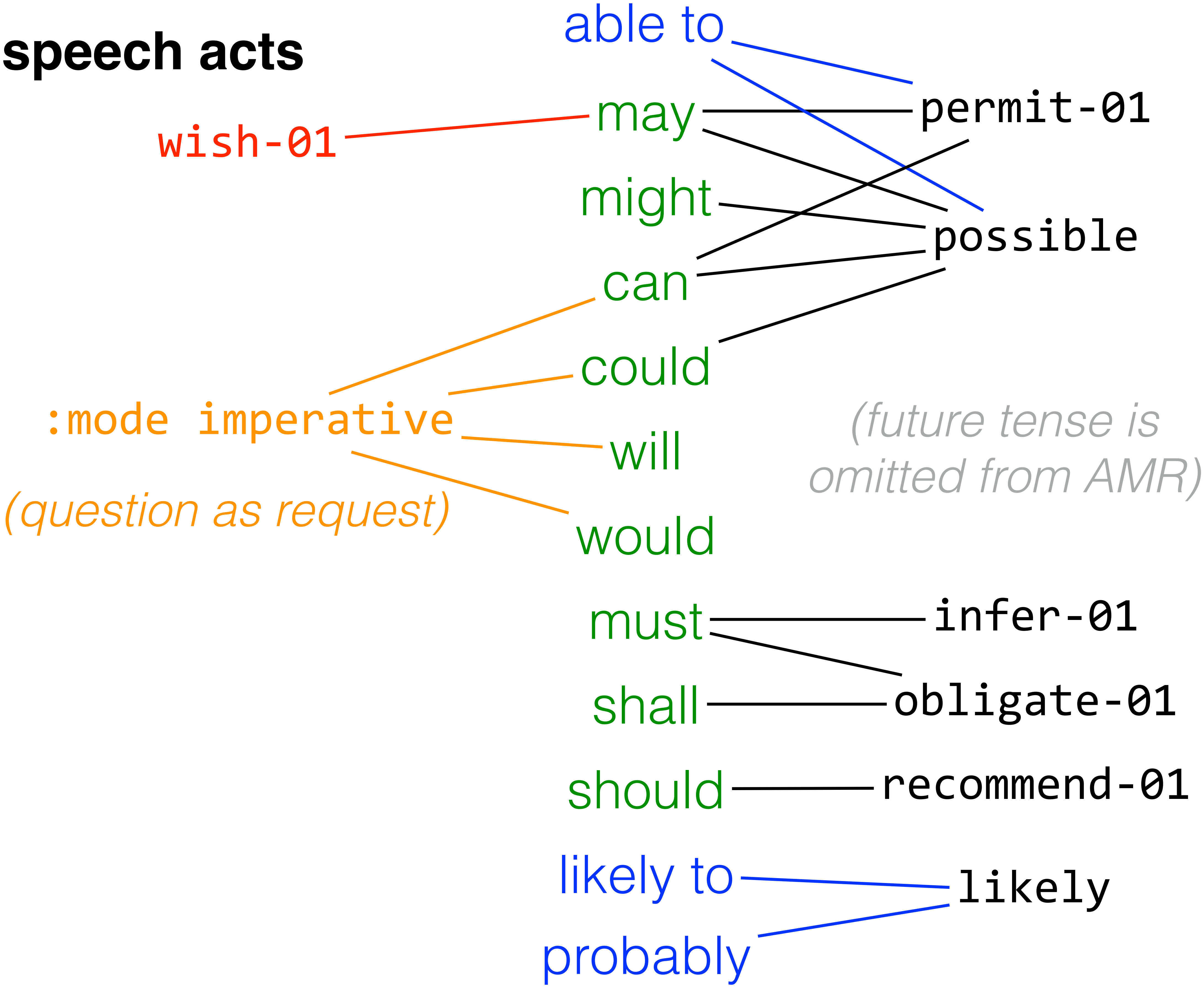
- You can leave.
  You may leave.
  It's all right for you to leave.

```
(p / permit-01
   :ARG1 (l / leave-01
            :ARG0 (y / you))
```

- I can see Russia from my house!
  I'm able to see Russia from my house!

```
(p / possible
   :domain (s / see-01
              :ARG0 (i / i)
              :ARG1 (c / country :name (n / name :op1 "Russia"))
              :location (h / house :poss i)))
```

speech acts


able to
may
might
can
could
will
would
must
shall
should
likely to
probably
wish-01
:mode imperative
(question as request)
permit-01
possible
(future tense is omitted from AMR)
infer-01
obligate-01
recommend-01
likely

Sentence Types

Type	AMR treatment
indicative (declarative)	*(default)*
imperative (command)	`:mode imperative` (with `you` arg if implied subject)
interjection	`:mode expressive`
yes-no question	`:mode interrogative`
WH-question	`amr-unknown`
quotation without speech verb	`(s / say-01 :ARG0 <speaker> …)`
vocative	`(s / say-01 :ARG2 <addressee> …)`
polite (“please”, etc.)	`:polite +`

Questions: yes-no

- Are you worried?

```
(w / worry-01
   :ARG0 (y / you)
   :mode interrogative)
```

Questions: wh

- Why worry? (What is the point of worrying?)

```
(w / worry-01
   :ARG0 (y / you)
   :purpose (a / amr-unknown))
```

Think of `amr-unknown` as an *in situ* question pronoun. Structurally, the AMR is the same as a declarative sentence.

- What's the problem?

```
(p / problem
   :domain (a / amr-unknown))
```

- How many peppers did Peter Piper pick?

```
(p / pick-10
   :ARG0 (p2 / person :name (n / name :op1 "Peter" :op2 "Piper"))
   :ARG1 (p3 / pepper
            :quant (a / amr-unknown)))
```

Comparison

have-degree-91
ARG1: attribute
ARG2: domain, entity characterized by attribute
ARG3: degree itself
ARG4: compared-to
ARG5: consequence, result of degree

- The treatment of comparative constructions is changing.
- Apples are redder than bananas.

old way:

```
(r / red
   :domain (a / apple)
   :degree (m / more)
   :compared-to (b / banana))
```

new way:

```
(r / red
   :ARG1-of (h / have-degree-91
         :ARG2 (a / apple)
         :ARG3 (m / more)
         :ARG4 (b / banana)))
```

Quantification

- two apples

```
(a / apple
   :quant 2)
```

- a lot of apples

```
(a / apple
   :quant (l / lot))
```

Only explicit quantifiers are included in the AMR.

- All apples are fruit.

```
(f / fruit
   :domain (a / apple
               :quant (a / all)))
```

- Apples are fruit.

```
(f / fruit
   :domain (a / apple))
```

Sets

include-91 - "subset"
ARG1: subset (or member)
ARG2: superset
ARG3: relative size of subset compared to superset

- Special predicate include-91 for explicitly mentioned sets

- (I ate) 5 of the 12 donuts

```
(d / donut :quant 5
   :ARG1-of (i / include-91
               :ARG2 (d2 / donut :quant 12)))
```

- 42% of the donuts

```
(d / donut
   :ARG1-of (i / include-91
               :ARG2 (d2 / donut)
               :ARG3 (p / percentage-entity :value 42)))
```

Reification

be-located-at-91 -
“reification of :location”
ARG1: entity
ARG2: location

- *the man at the store*

  (m / man :location (s / store))

- What about: *the man* ***always*** *at the store*?
  - Need to “modify” the relation!
  - Solution: Convert (“**reify**”) the relation w/ a special frame

```
(m / man
    :ARG1-of (b / be-located-at-91
                  :ARG2 (s / store)
                  :time (a / always)))
```

Reification

be-located-at-91 -
"reification of :location"
ARG1: entity
ARG2: location

- Reification also allows a relational predicate to be focused.

- The man **is** at the store.

```
(b / be-located-at-91
    :ARG1 (m / man)
    :ARG2 (s / store))
```

- I think the man **is** at the store.

```
(t / think-01
   :ARG0 (i / i)
   :ARG1 (b / be-located-at-91
                :ARG1 (m / man)
                :ARG2 (s / store)))
```

Reification

be-located-at-91 - “reification of :location”
ARG1: entity
ARG2: location

- Every role has a designated reification—either a verb frame or a special -91 frame.
  - have-purpose-91, have-polarity-91, have-part-91, …
  - ~~have-topic-91~~ concern-02
- These slides **are** about semantics.

```
(c / concern-02
    :ARG0 (s / slide :mod (t / this))
    :ARG1 (s2 / semantics))
```

English is a wily opponent.

- Copulas, light verbs
- Prepositions
- Derivational morphology
- Relational nouns
- Coordination & clausal connectives
- Modality
- Non-declarative sentences
- Questions
- Comparisons
- Quantification
- Subsets
- Reification

Other Phenomena

- Many other patterns for specific phenomena are documented in the **AMR Dictionary**. E.g.:
  - *We'll eat **like** kings* → resemble-01
  - *(Banarescu et al., 2013)* → publication-91

AMR Dictionary

how

- :ARGx
  - **How much** does this cost? `(cost-01 :ARG2 amr-unknown)` Example
  - **How** did you solve the problem? `(solve-01 :ARG2 amr-unknown)` Example
- :degree
  - **How** big was the dog that bit you? `(big :degree amr-unknown :domain :dog)` Example
  - **How beautiful** you are. [= You are so beautiful.] `(beautiful :degree so :domain you)` Example
- :manner
  - **How** did you get him to help you? `(get-04 :manner amr-unknown)` Example
  - The only thing that surprises me is **how rapidly** this is happening. `(surprise-01 :ARG0 (rapid :manner-of this))` Example
- :quant
  - **How tall** are you? `(tall :quant amr-unknown :domain you)` Example
  - **How old** is your father? `(father :age amr-unknown :poss you)` Example

however

- contrast-01 (most cases)
  - John, however, stayed at home. `(contrast-01 :ARG2 stay-01)` Example
- :concession
  - It started to rain; however, the game continued. `(start-01 :ARG1 rain-01 :concession-of continue-01)` Example

imperative

- :mode imperative
  - Go home! `(go-02 :mode imperative :ARG0 you :destination home)` Example
  - Let's go home. `(go-02 :mode imperative :ARG0 we :destination home)` Example
  - Come on folks!! `(come-25 :mode imperative :ARG1 folk)` Example
- :mode imperative :polite +
  - **Please** close the door. `(close-01 :mode imperative :polite +)` Example
  - **Could** you close the door? `(close-01 :mode imperative :polite +)` Example

Further AMR Documentation

- **Homepage:** http://amr.isi.edu/
- **Guidelines:** https://github.com/amrisi/amr-guidelines/blob/master/amr.md
- **Editor help pages**, especially AMR Dictionary (http://www.isi.edu/~ulf/amr/lib/amr-dict.html)
- **Editor release search** (`rs` *`query`*) to check existing AMRs for a precedent

AMR Data

Real Data

- Thus far: mostly made-up examples
- Real sentences tend to be longer, but AMRed using the same principles

Real Data

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

Real Data

control structure

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

Real Data

subset → include-91

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

Real Data

temporal connective

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

Real Data

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

modal

Real Data

- We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

deverbals → event predicates

Real Data

We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

```
(n / need-01
      :ARG0 (w / we)
      :ARG1 (b / borrow-01
            :ARG0 w




                                              )







                                                     )
```

Real Data

We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

```
(n / need-01
    :ARG0 (w / we)
    :ARG1 (b / borrow-01
        :ARG0 w
        :ARG1 (p / percentage-entity :value 55
            :ARG1-of (i / include-91
                :ARG2 (p2 / price
                    :mod (h / hammer))))
```

)

Real Data

We need to borrow 55% of the hammer price until we can get planning permission for restoration which will allow us to get a mortgage.

```
(n / need-01
    :ARG0 (w / we)
    :ARG1 (b / borrow-01
        :ARG0 w
        :ARG1 (p / percentage-entity :value 55
            :ARG1-of (i / include-91
                :ARG2 (p2 / price
                    :mod (h / hammer))))
        :time (u / until
            :op1 (p3 / possible
                :domain (g / get-01
                    :ARG0 w
                    :ARG1 (p4 / permit-01
                        :ARG1 (p5 / plan-01)
                        :purpose-of (r / restore-01)
                        :ARG0-of (a / allow-01
                            :ARG1 (m / mortgage-01
                                :ARG0 w))))))))
```

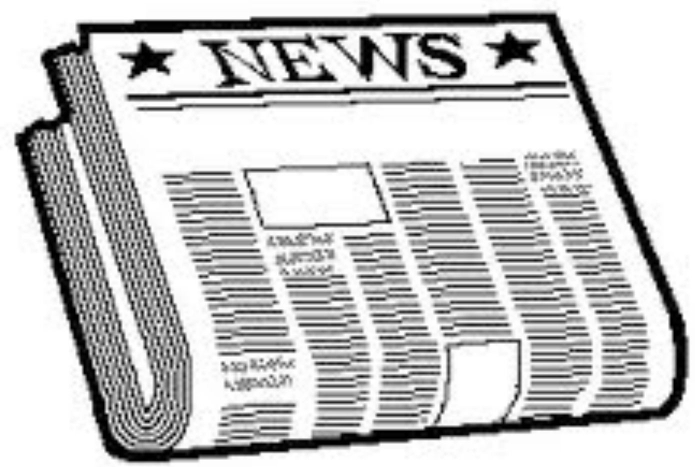
Datasets

AMR Bank: *The Little Prince*

(novel—English translation)

LDC Releases

(news, discussion forums, …)



+

= 15k AMRs

(with more to come!)

Datasets: Details

- AMR Bank (Release 1.4; http://amr.isi.edu/download/amr-bank-v1.4.txt)
- English translation of *Le Petit Prince (The Little Prince)*, freely downloadable. 1,500 AMRs
- AMR Public Release 1.0 (LDC2014T12): largest public release w/ 13,051 AMRs
- DEFT Release 3 (LDC2013E117): evaluation data in Flanigan et al. 2014, Wang et al. 2015.
- DEFT Release 4 (LDC2014E41): largest release w/ 18,779 AMRs total
- DEFT Release 5 (Sep. 2015) will include wikification, (pretty much) no directed cycles
- Small (100-AMR) sets of Czech and Chinese AMRs have been annotated.
- Vanderwende et al. (2015) data to appear: several languages, automatically converted from logical forms
- PropBank will soon all be converted to AMR style (mapping nominalizations to verbs, etc.) and re-released.

Working with AMR Data

- AMRs are stored in a plain text format (in Penman notation)
- Script for loading them into a Python data structure: https://github.com/nschneid/amr-hackathon/tree/master/src
  - Also accepts **aligned AMRs** from the ISI aligner

AMR
vs.
Other Formalisms

AMR Strengths

- **abstracting away from morphological & syntactic variability**
- predicate-argument structures
  - core + non-core roles
- named entities & values
- coreference (w/in sentence)
- modality

AMR Limitations

- no “deep” **lexical semantics**
  - *fruit*/*berry*, *buy*/*sell*, *kill*/*die* are formally unrelated
- no deep treatment of **quantification & scope**
- (almost) no **information structure**
- nothing across sentences in a **discourse**…yet

Design Decisions

- AMR annotations are **not tied to individual words or any syntactic derivation**
- **Practicality for human annotators** is primary
  - AMR makes no compromises for (current) algorithms
- **Single structure** rather than many layers
- Extensive **documentation and tool support**

Detailed Comparison

- See “Other Formalisms” slides

Comparison - Semantic Roles

AMR: 70+ non-core roles, many verb-sense specific roles (up to 5 args/roleset, more than 10,000 rolesets)

FrameNet: large inventory of frame-specific roles

VerbNet: inventory of thematic roles

Groningen Meaning Bank: VerbNet inventory

Most others: small inventory of roles (agent, theme, etc.)

Comparison - Sense Lexicon

Groningen Meaning Bank: (automatic) WordNet synsets

FrameNet/UCCA: Mark senses by frame/script, not lemma

AMR /PropBank: coarse-grained senses (get high ITA)

Prague Dependency TB: valency lexicon rolesets

Most others: undisambiguated concepts as predicates

Comparison - Entities

AMR: Rich named entity ontology (100+ types), wikification

GALE/Ontonotes Annotations: 29 types, 64 subtypes

Groningen Meaning Bank: 7 NE types

Domain-specific (ACE/UMLS/etc.): rich; not all entities

Others: no entity typing

Comparison - Alignment with text

Deepbank; Groningen Meaning Bank: Semantics linked up to a theory of its derivation from syntax (HPSG; CCG)

PropBank, Semantic Treebank: grounded in PTB

Most others: Some link to words in sentence

AMR: No alignment to text (plan to release a few alignments)

Comparison - Logic/Scope/Entailments

Deepbank; Groningen Meaning Bank: Semantics grounds out in logical formalisms (DRT and MRS, respectively)

AMR entailment: linkage between its lexicon and VerbNet may allow rich decomposition

AMR scope: No scope of quantification

Comparison - Size and Quality

AMR: 18,779 sentences, goes beyond newswire, fully manual

Prague Dependency TB: WSJ in Czech and English, manual

Deepbank; Groningen Meaning Bank: Large; automatic parses with human correction/feedback.

UCCA: fully manual, 160k tokens

Rich semantic systems with little affiliated data: TMR, LCS, ...

Part II:
Algorithms and Applications

Speaker: Jeffrey Flanigan

Intro

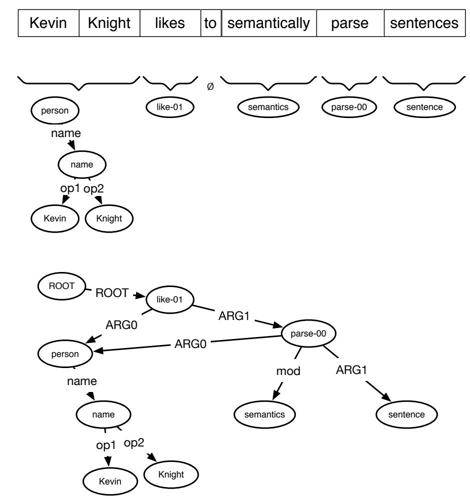
- You know how to annotate AMRs
- Now, we want to use them!
- To use AMRs, we need automatic **parsers**
- But first: **alignment**
- **Evaluation** (inter-annotator agreement, parser output)
- And also:
  - **Graph grammars** (like CFGs, but for graphs)
  - **Applications**

Alignment

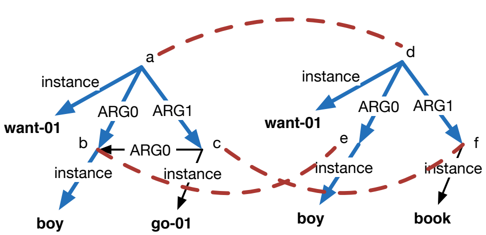
IAEA accepted North Korea 's proposal in November.

```
(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))
```

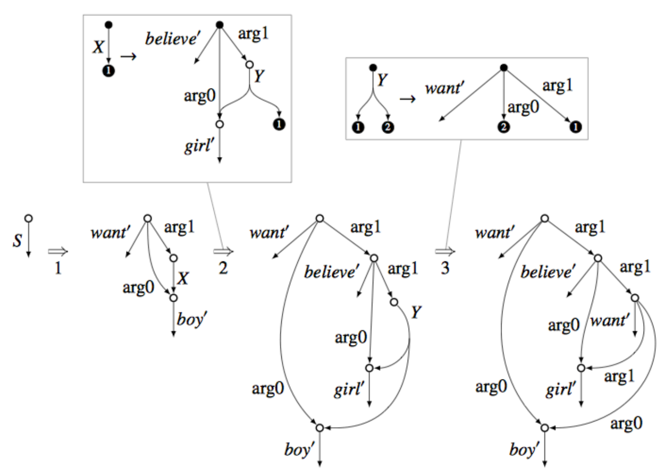
Parsing



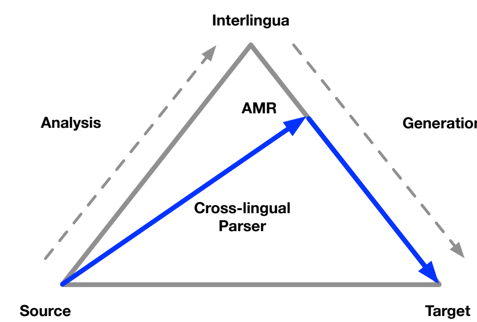
Evaluation



Graph grammars



Applications



Outline

- Alignment
- Parsing
- Evaluation
- Graph Grammars and Automata
- Applications

Alignment: Motivation

- AMR annotation has no explicit alignment to sentence
- Training data has whole sentence – AMR graph pairs
- For generalization performance, need fine-grained correspondence between words and pieces of AMR
- Alignments provide this correspondence

Need alignments to train parsers, etc

Alignment

The **tour** was a **surprise offer** made ***by*** **North Korea** ***in*** **November**.

```
(t / thing
     :ARG0-of (s / surprise-01)
     :ARG1-of (o / offer-01
          :ARG0 (c / country
               :name (n / name
                    :op1 "North"
                    :op2 "Korea"))
          :time (d / date-entity
                    :month 11))
     :domain (t2 / tour-01))
```

- Align concepts with words
- Can also align edges with function words

Alignment

- Alignment
  - Motivation
  - **JAMR's rule-based aligner**
  - ISI EM aligner
- Parsing
- Evaluation
- Graph Grammars and Automata
- Applications

JAMR Aligner (Flanigan et al, 2014)

- Aligns graph fragments to spans of words (edges not in fragments are unaligned)
- Uses a set of handcrafted rules
- Uses lemmatizer, string edit distance to match concepts with words
- Rules for: named entities, date entities, special concepts, negation, degrees, etc (15 total rules)

JAMR Aligner

For each rule

- Greedily align concepts in a depth first traversal of the AMR graph
- Rules are applied in a specified order

JAMR Aligner

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))
```

JAMR Aligner

Rule 1) Date entity

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))
```

JAMR Aligner

Rule 3) Named entity

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))
```

JAMR Aligner

Rule 5) Single concept (use lemma)

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))
```

JAMR Aligner

Rule 6) Fuzzy single concept (longest string prefix > 4)

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))
```

JAMR Aligner

Rule 10) person-of/thing-of

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))
```

JAMR Aligner

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))
```

JAMR Aligner

Evaluate on 200 hand-aligned sentences:

F_1: 90%   Precision: 92%   Recall: 89%

<u>Extracted concept table</u>

```
8       critical => (critical)
2       critical => (criticize-01)
1       critically => (critical)
1       criticised => (criticize-01)
14      criticism => (criticize-01)
30      criticized => (criticize-01)
1       critics => (critic)
4       critics => (person :ARG0-of (criticize-01))
5       crop => (crop)
5       crops => (crop)
3       cross => (cross)
15      cross => (cross-02)
3       cross => (cross-border)
2       cross => (cross-strait)
1       crossed => (cross-00)
2       crossing => (cross-02)
```

ISI Aligner (Pourdamghani et al, 2014)

- Aligns each concept or edge to at most one word
- Learns from data using EM
- Inspired by MT alignment models
- Basic idea: convert graph to linear string, use word alignment model

ISI Aligner

IAEA accepted North Korea 's proposal in November.

```
(a / accept-01
    :ARG0 (o / organization
        :name (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))
```

ISI Aligner

IAEA accepted North Korea ’s proposal in November.

```
accept-01 :ARG0 organization :name name :op1
“IAEA” :ARG1 thing :ARG1-of propose-01 :ARG0
country :name name :op1 “North” :op2 “Korea” :time
date-entity :month 11
```

Linearize the AMR using a depth-first traversal

ISI Aligner

IAEA accepted North Korea proposal in November

```
accept organization name IAEA thing propose-01
country name North Korea :time date-entity 11
```

English: remove stop words

AMR: remove special concepts, relations that don't usually align, quotes, and sense tags

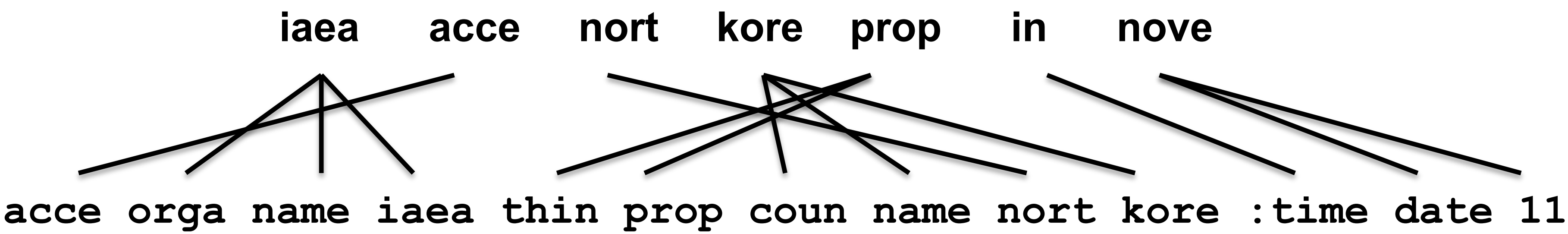
ISI Aligner

iaea acce nort kore prop in nove

`acce orga name iaea thin prop coun name nort kore :time date 11`

Both: Lowercase and stem to the first four letters

ISI Aligner



Run IBM alignment models with a symmetrization constraint, and project to AMR graph

Alignments are 1-to-many

Alignment: Summary

	JAMR aligner	**ISI aligner**
Alignment type	Graph fragment to span of words	Each concept or edge to at most one word
Aligns edges	No	Yes
Learned from data	No	Yes
Gold standard available	https://github.com/jflanigan/jamr	http://amr.isi.edu/research.html
F_1 score on concepts*	90% (spans)	89.8%
F_1 score on relations*	NA	49.3%

*JAMR and ISI not directly comparable, since different gold standard

In general, the desired type of alignment will depend on the application

Parsing

- Alignment
- **Parsing**
  - Graph-based parsing
    - Structured prediction
    - Concept identification
    - Relation identification
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
    - Experiments
  - Transition-based parsing
  - Parsing using syntax-based MT
- Evaluation
- Graph Grammars and Automata
- Applications

Parsing

Kevin Knight likes to semantically parse sentences.

```
(l / like-01
     :ARG0 (p / person
          :name (n / name
               :op1 "Kevin"
               :op2 "Knight"))
     :ARG1 (p2 / parse-00
          :ARG0 p
          :ARG1 (s / sentence)
          :mod (s2 / semantics)))
```

JAMR Overview (Flanigan et al, 2014)

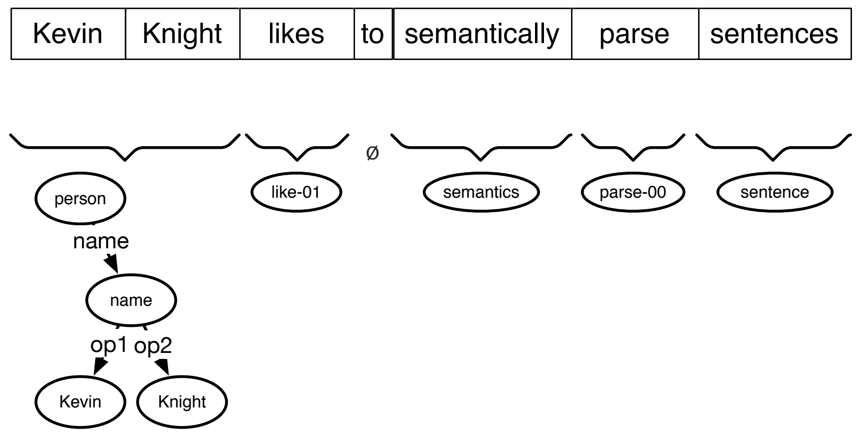
Input

Kevin	Knight	likes	to	semantically	parse	sentences

JAMR Overview

Input

Concept ID

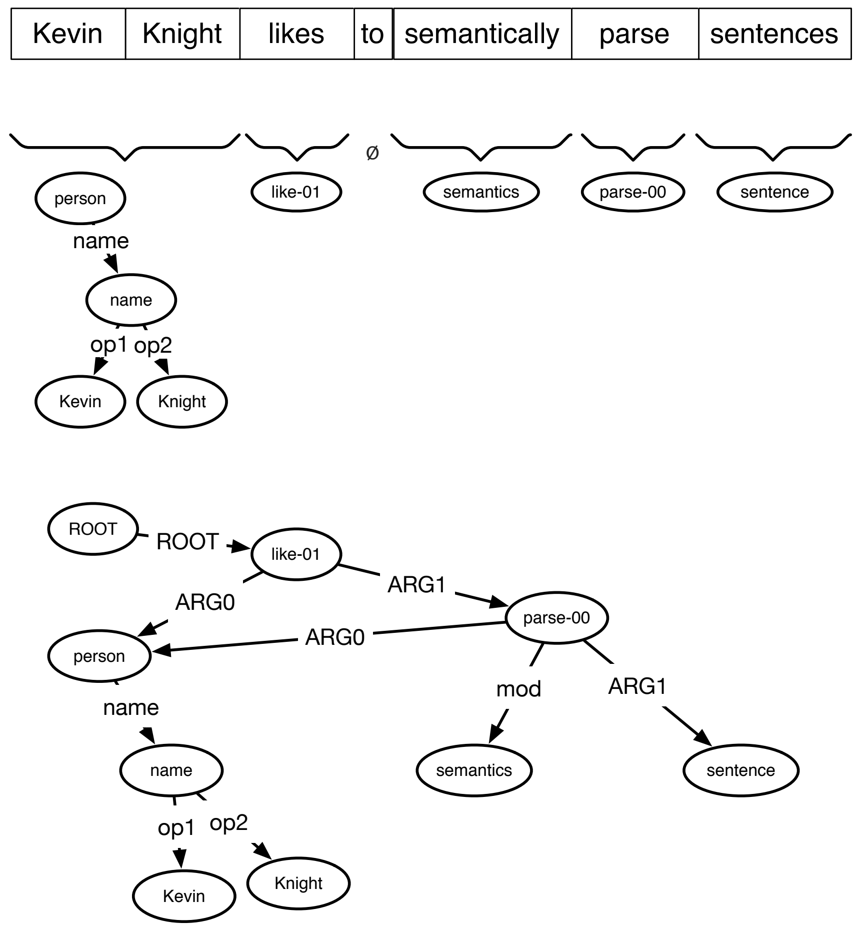


JAMR Overview

Input

Concept ID

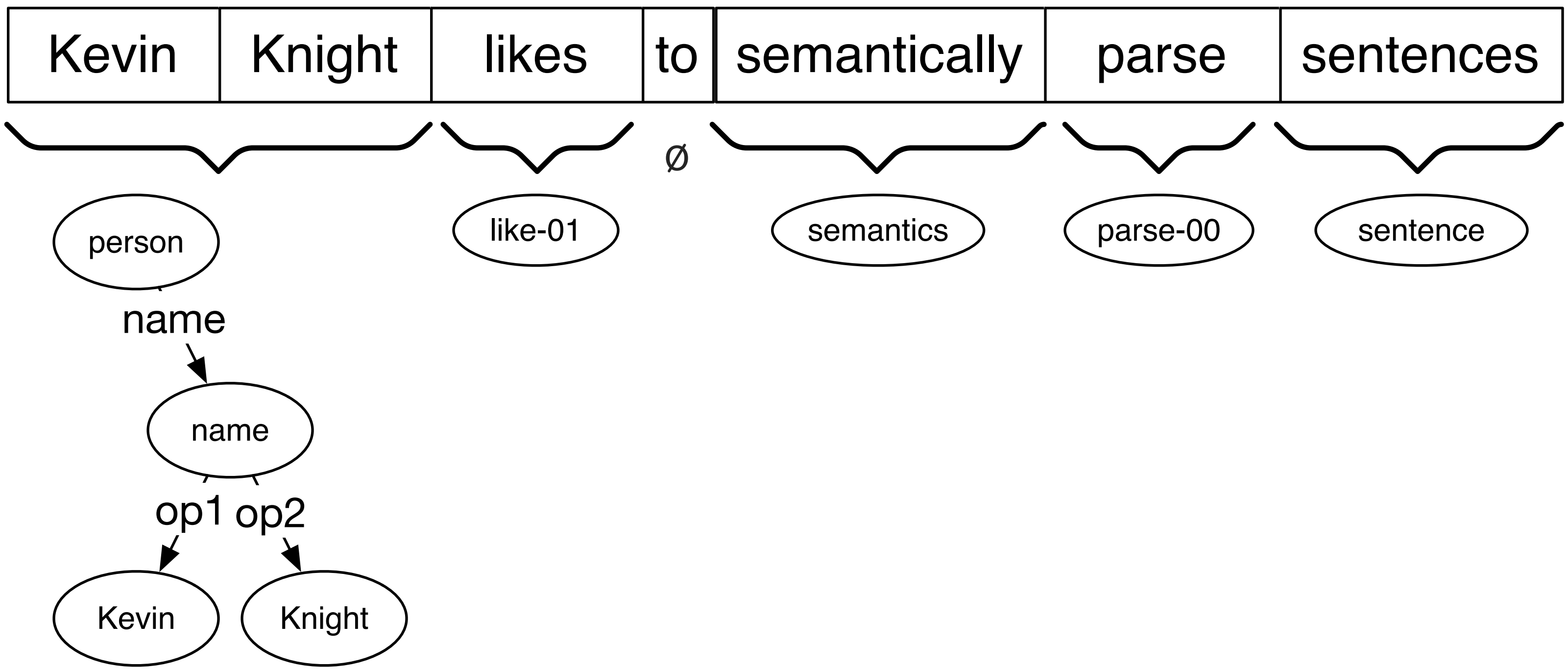
Relation ID



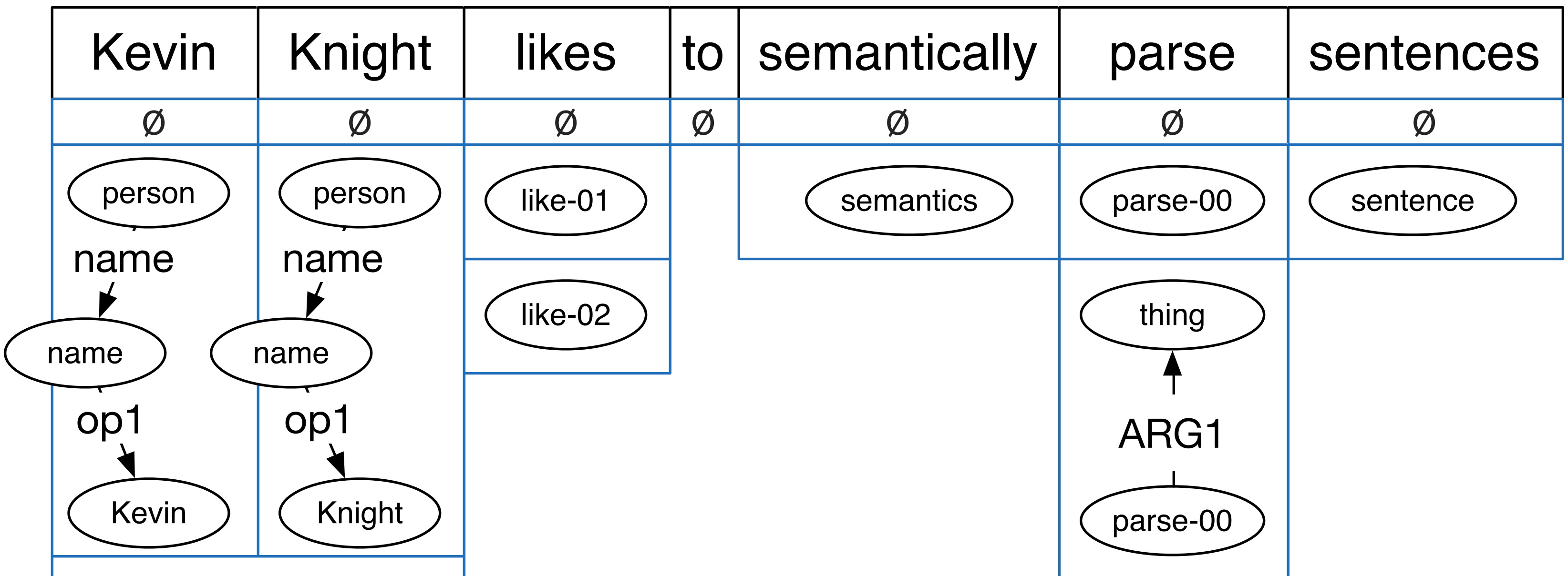
Concept Identification

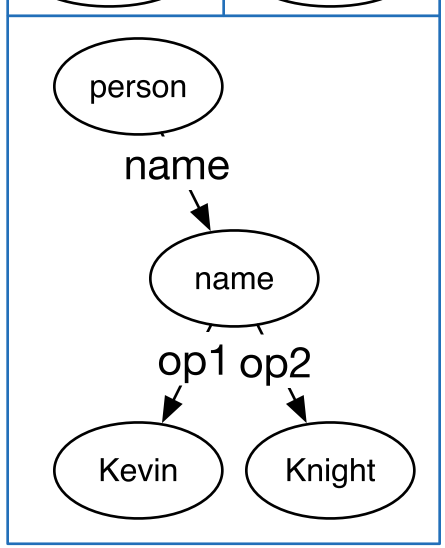
Kevin	Knight	likes	to	semantically	parse	sentences

Concept Identification



Concept Identification

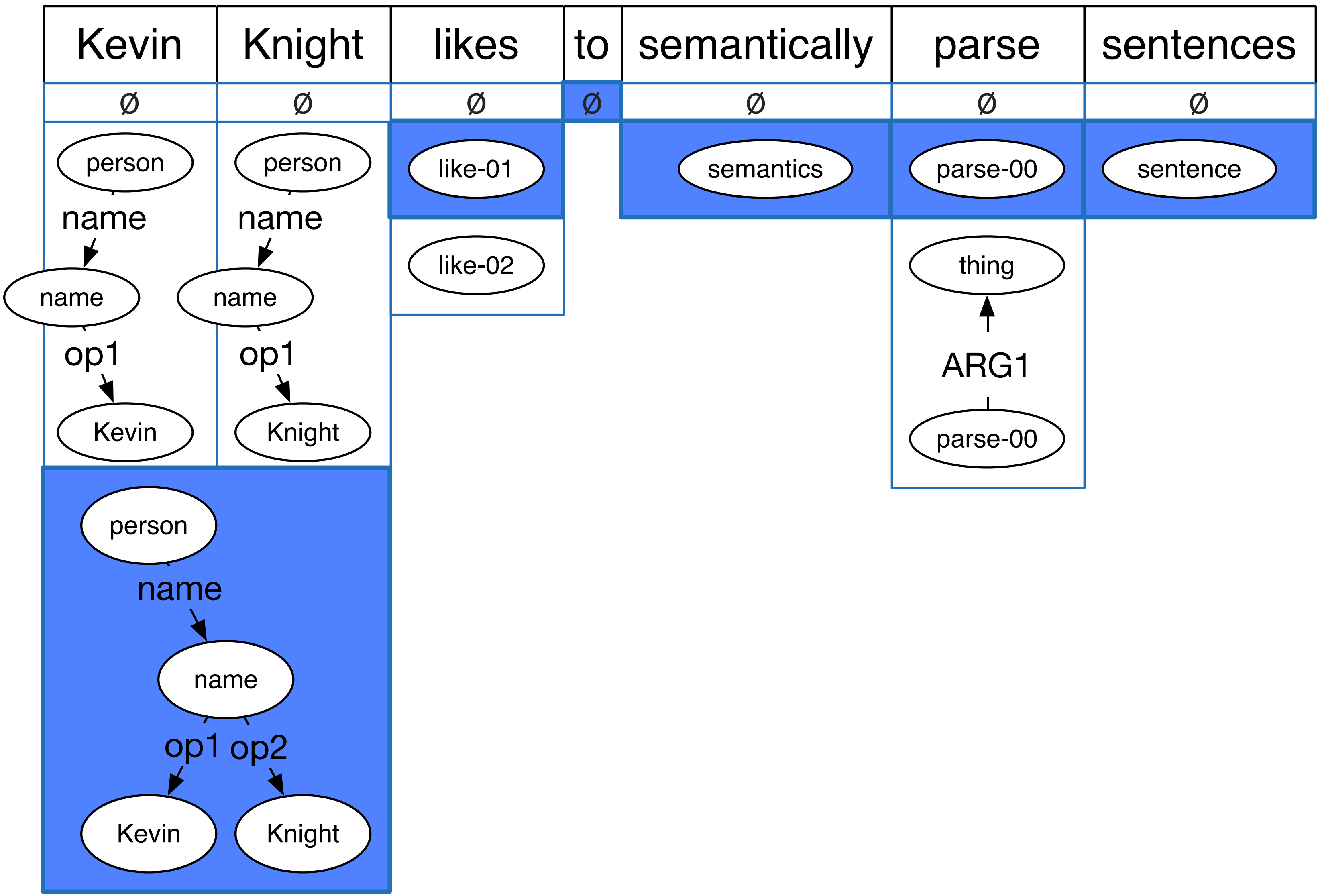




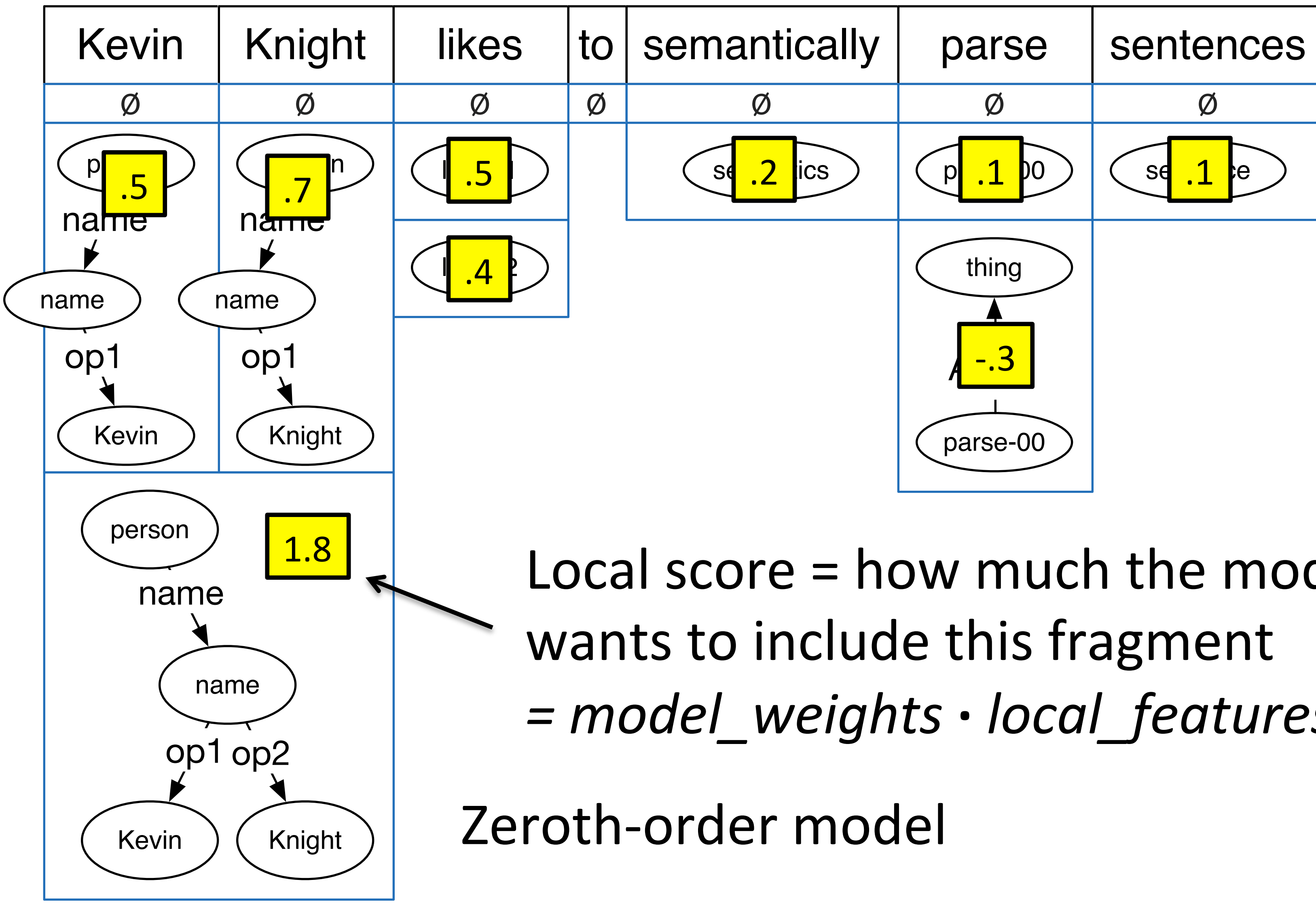
Extracted concept table

```
7     likes => like-01
1     likes => like-02
2     parse => parse-00
1     parse => (thing :ARG1-of (parse00)
1     semantically => semantics
2     sentences => sentence
```

Concept Identification



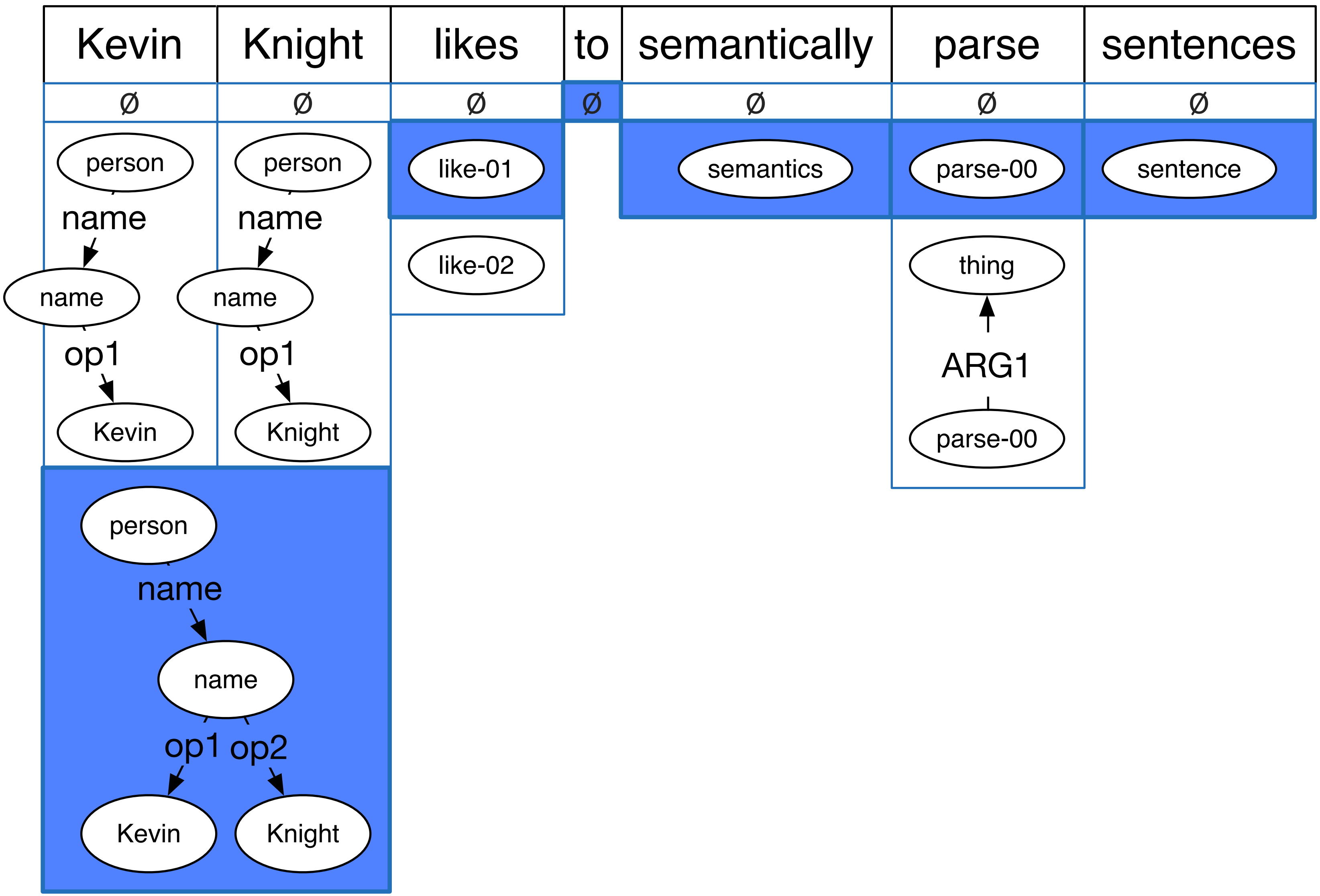
Concept Identification



Local score = how much the model wants to include this fragment
= *model_weights · local_features*

Zeroth-order model

Concept Identification



Training

- AdaGrad structured perceptron

Learning rate

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_i^{t'}}} g_i^t$$

Model weight component i at step t+1

Gradient

Training

- AdaGrad structured perceptron

Learning rate

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_i^{t'}}} g_i^t$$

Model weight component i at step t+1

Gradient

$$g_i^t = feat_i(X_t, \hat{Y}_t) - feat_i(X_t, Y_t)$$

Input

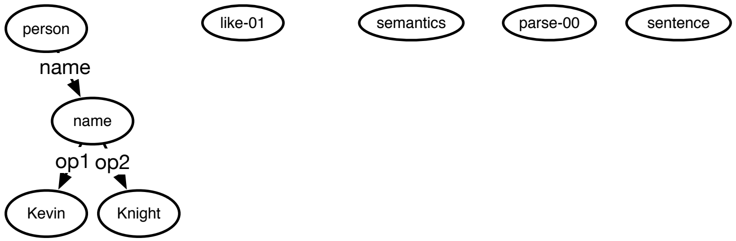
Predicted output

Gold output

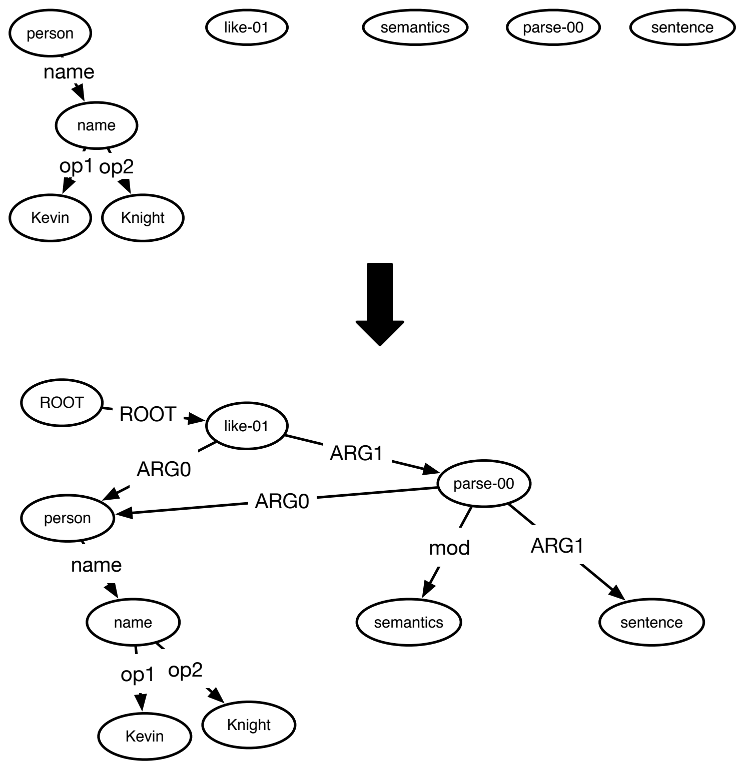
Relation Identification

- Evaluation
- Alignment
- Parsing
  - Graph-based parsing
    - Concept identification
    - **Relation identification**
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
    - Experiments
  - Transition-based parsing
  - Parsing using syntax-based MT
- Graph Grammars and Automata
- Applications

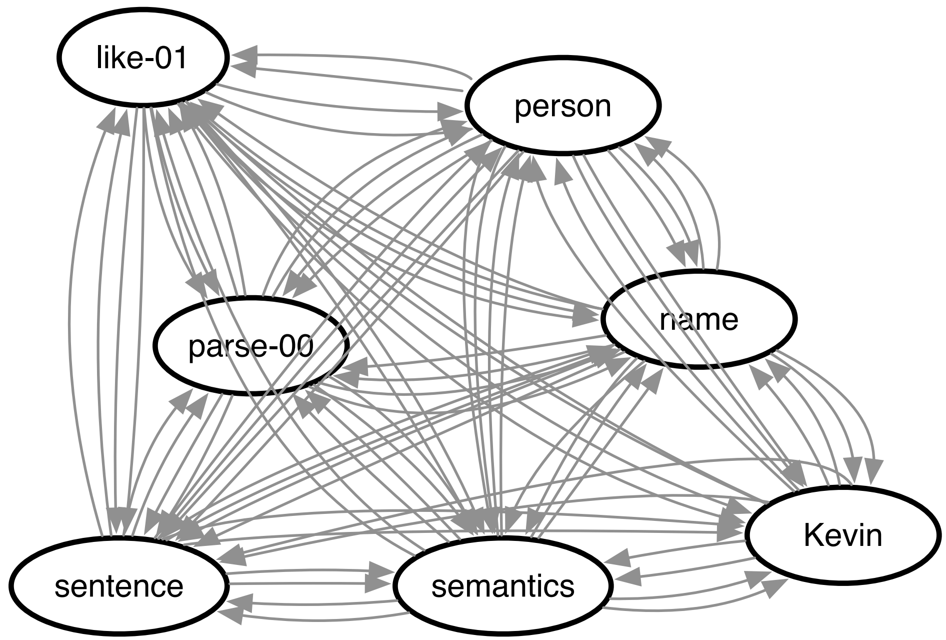
Relation Identification



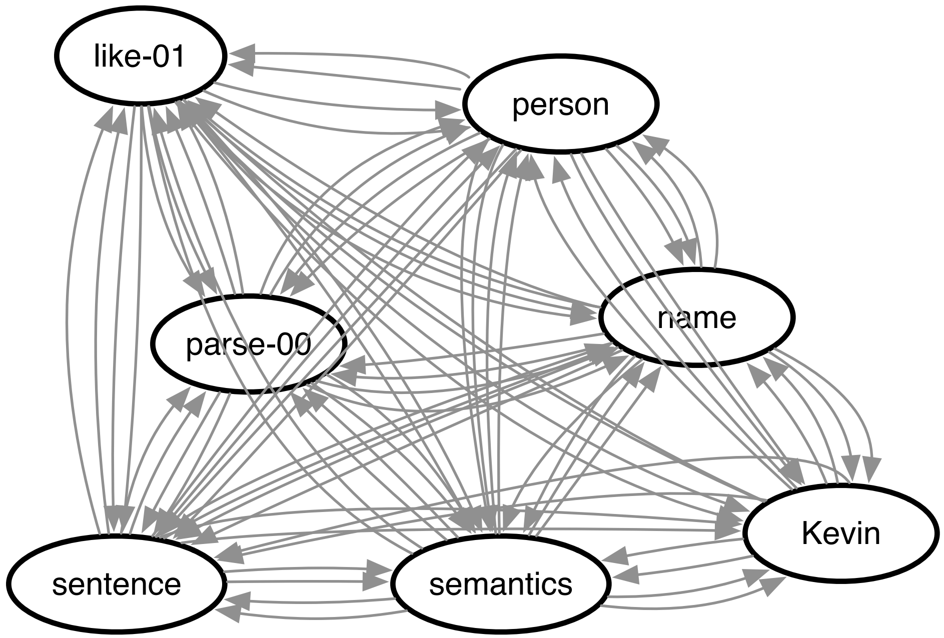
Relation Identification



Dense Graph

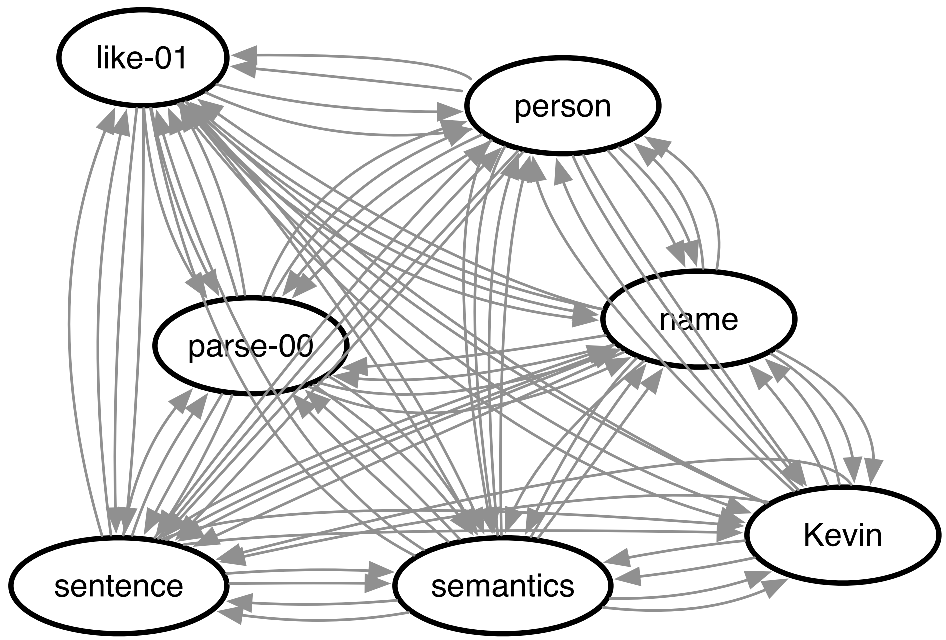


Dense Graph



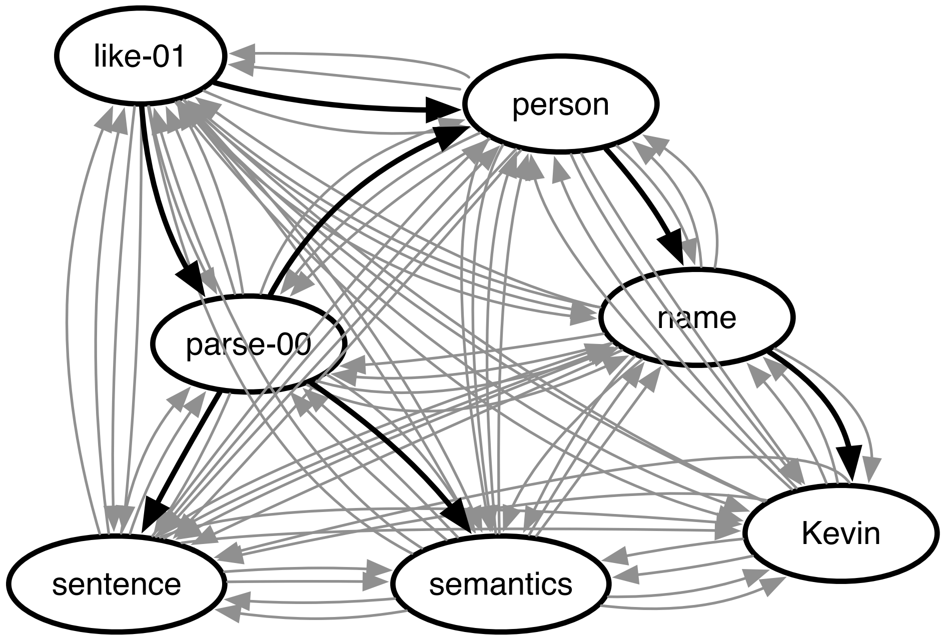
- All possible edges between all nodes
- Edges w/ weights

Dense Graph



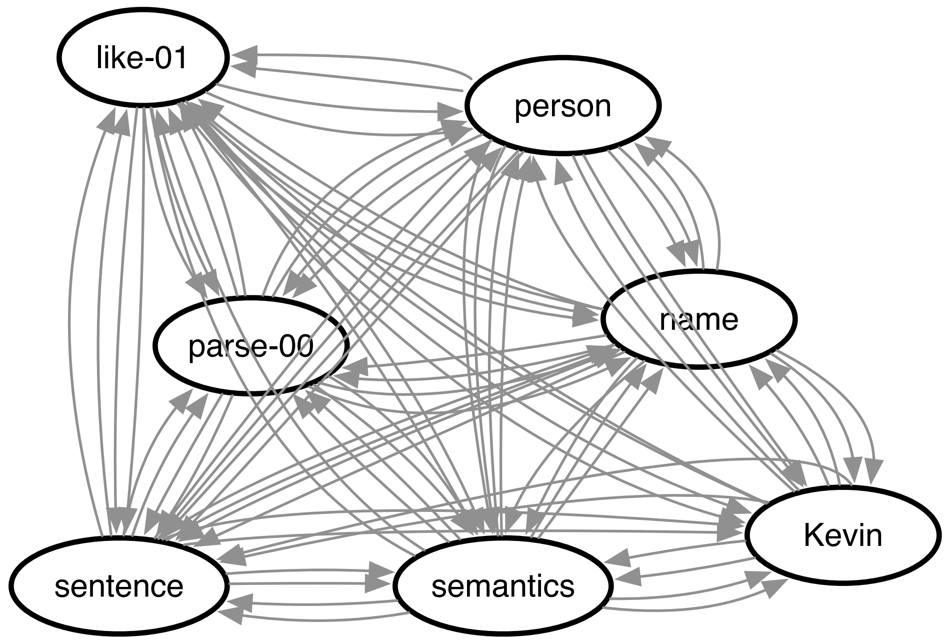
Edge weight = how much the model wants to include that edge in the output graph

Dense Graph



Output graph = max subgraph with constraints on well-formedness

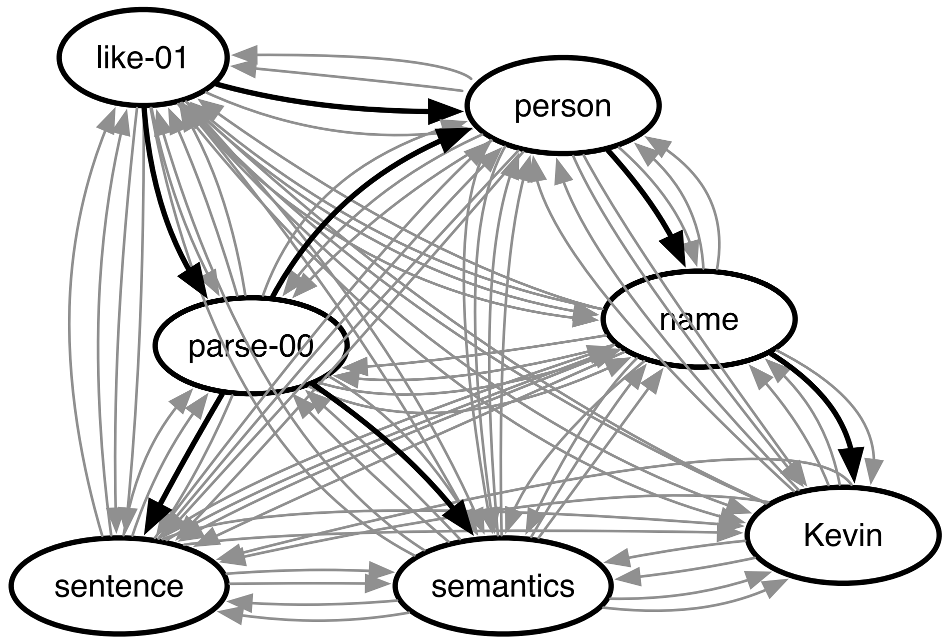
Dense Graph



$\mathbf{z}$ binary vector, indicates which edges are selected

ϕ real vector, contains the edge weights

Max Subgraph



Relation ID optimization problem

$$\max_{\mathbf{z}\in G} \phi^T \mathbf{z}$$

Set of graphs satisfying the constraints

Output Graph Properties (Constraints)

- Preserving
- Simple
- Spanning (all nodes)
- Connected
- Deterministic

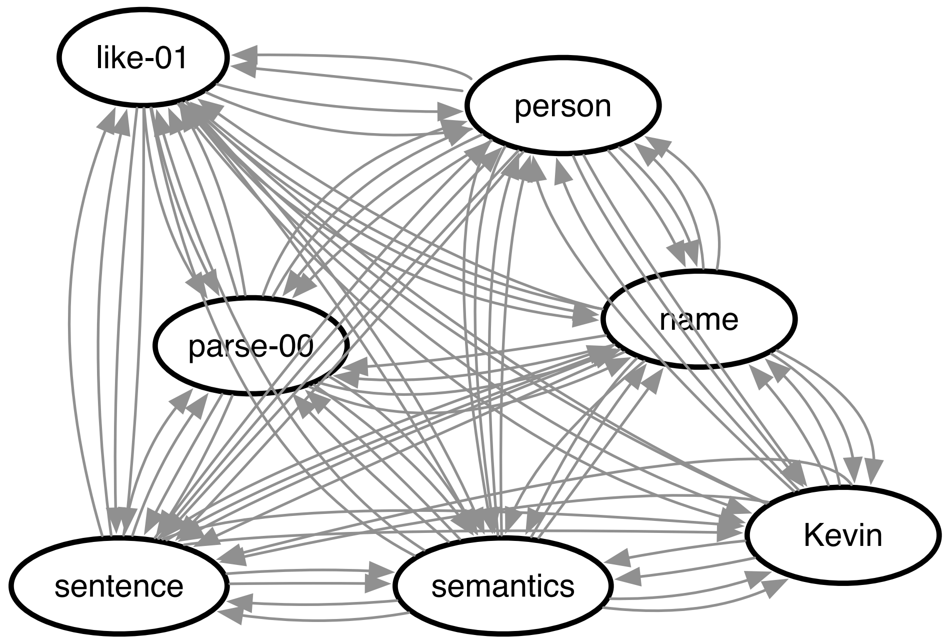
Output Graph Properties (Constraints)

$$\boxed{\begin{array}{l}\bullet\ \text{Preserving}\\ \bullet\ \text{Simple}\\ \bullet\ \text{Spanning (all nodes)}\\ \bullet\ \text{Connected}\end{array}}\ \Bigg\}\ \text{set } \mathcal{Z}$$

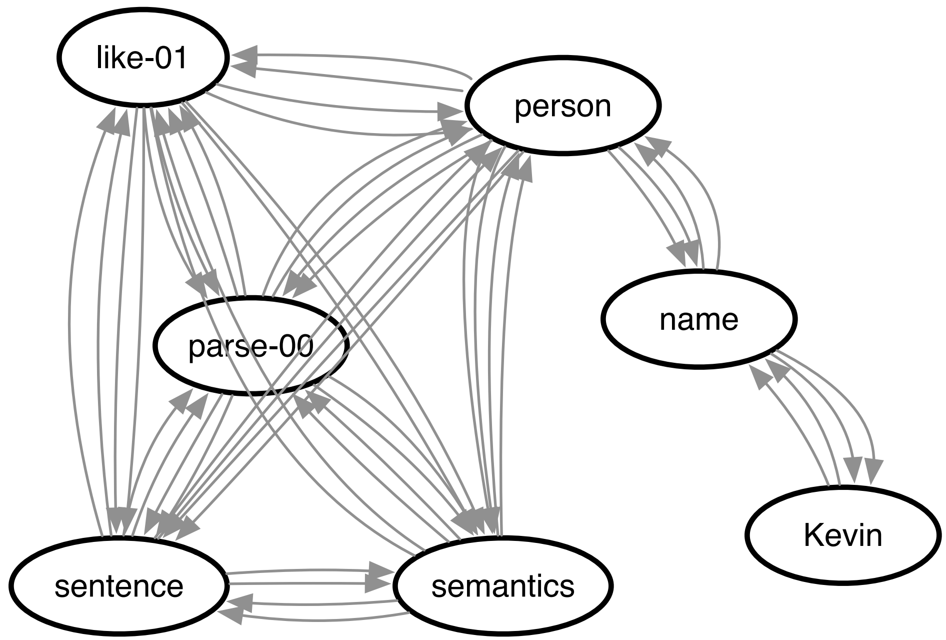
- Deterministic

The full list: $\Bigg\}$ set G

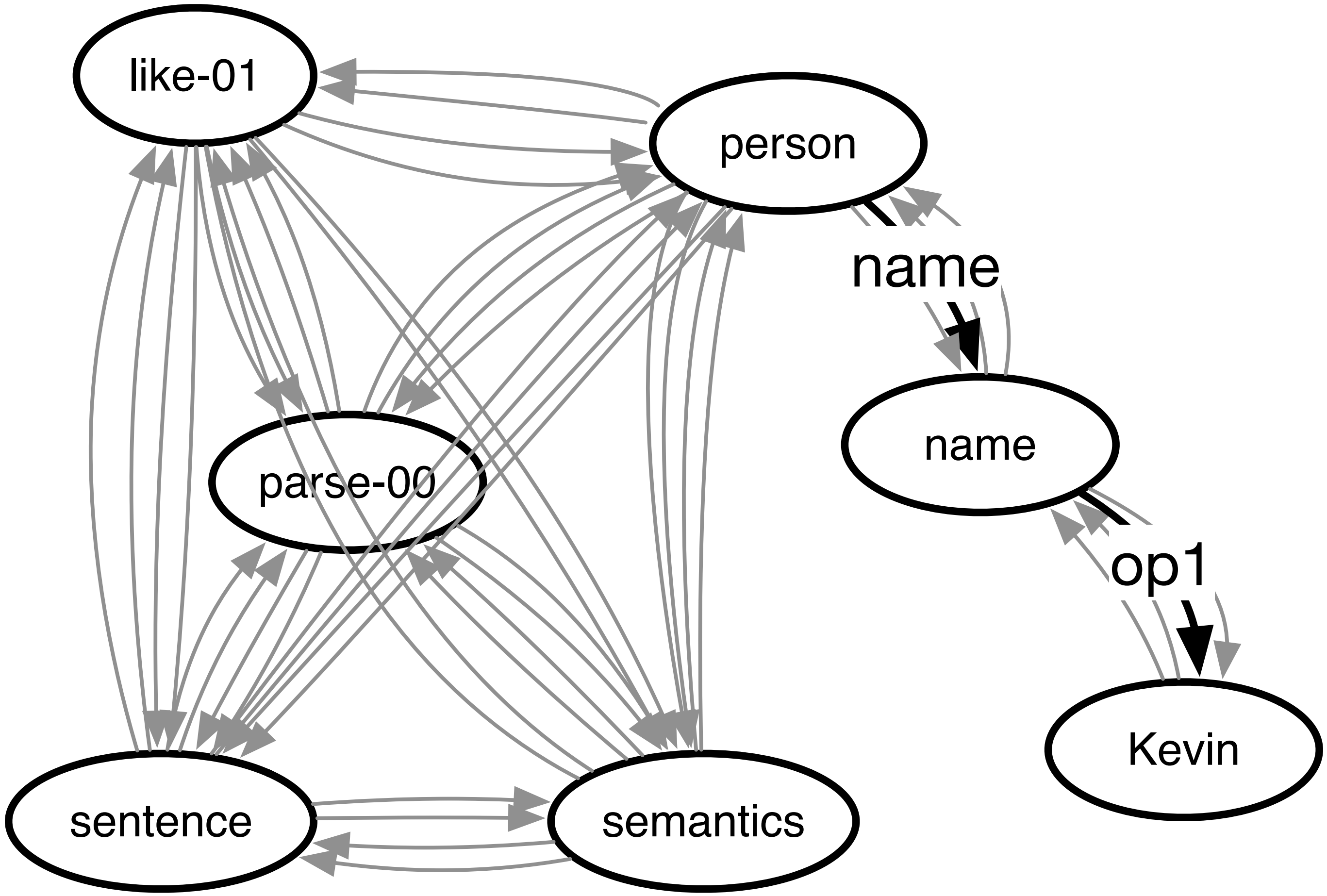
Dense Graph



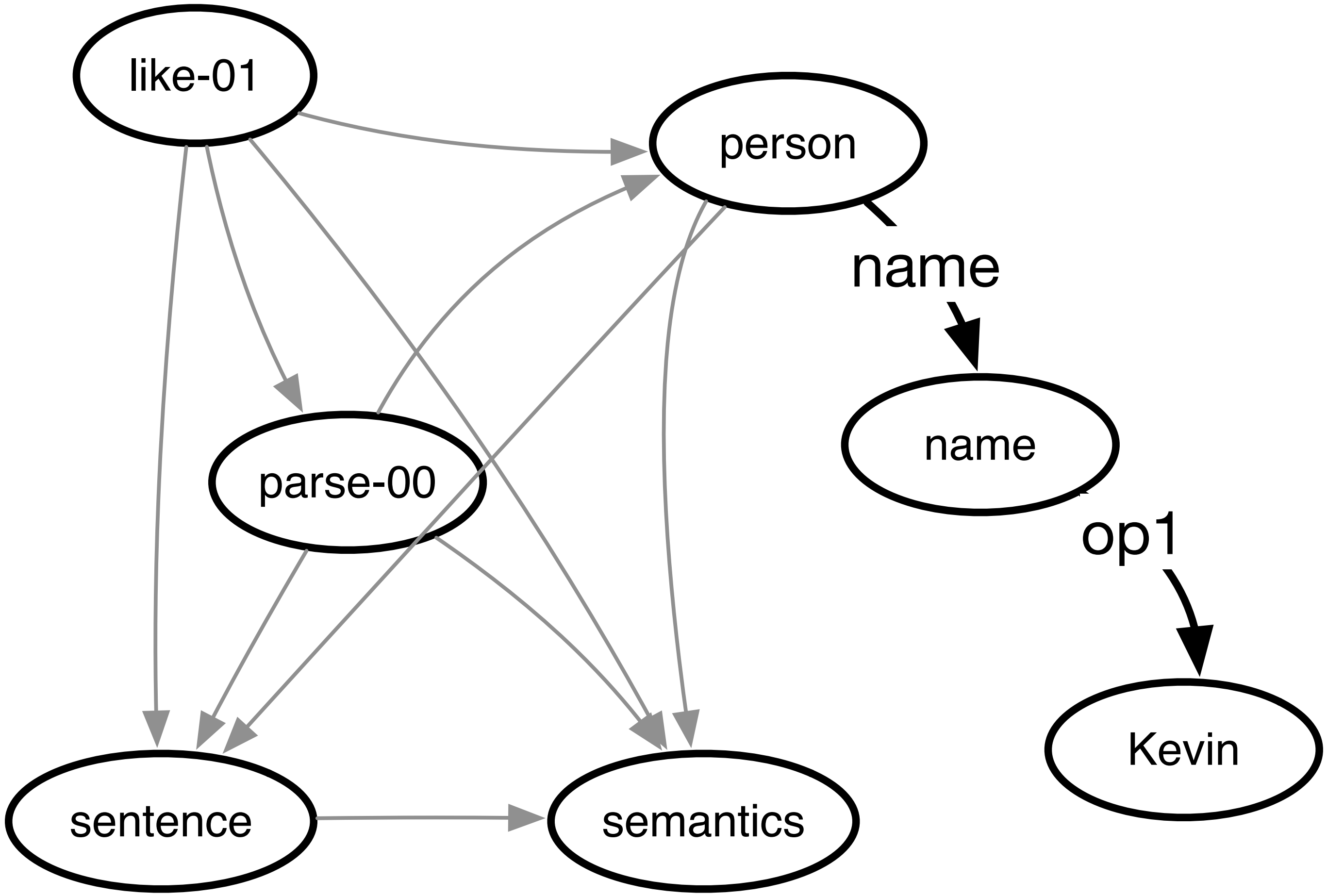
Reduced graph for clarity



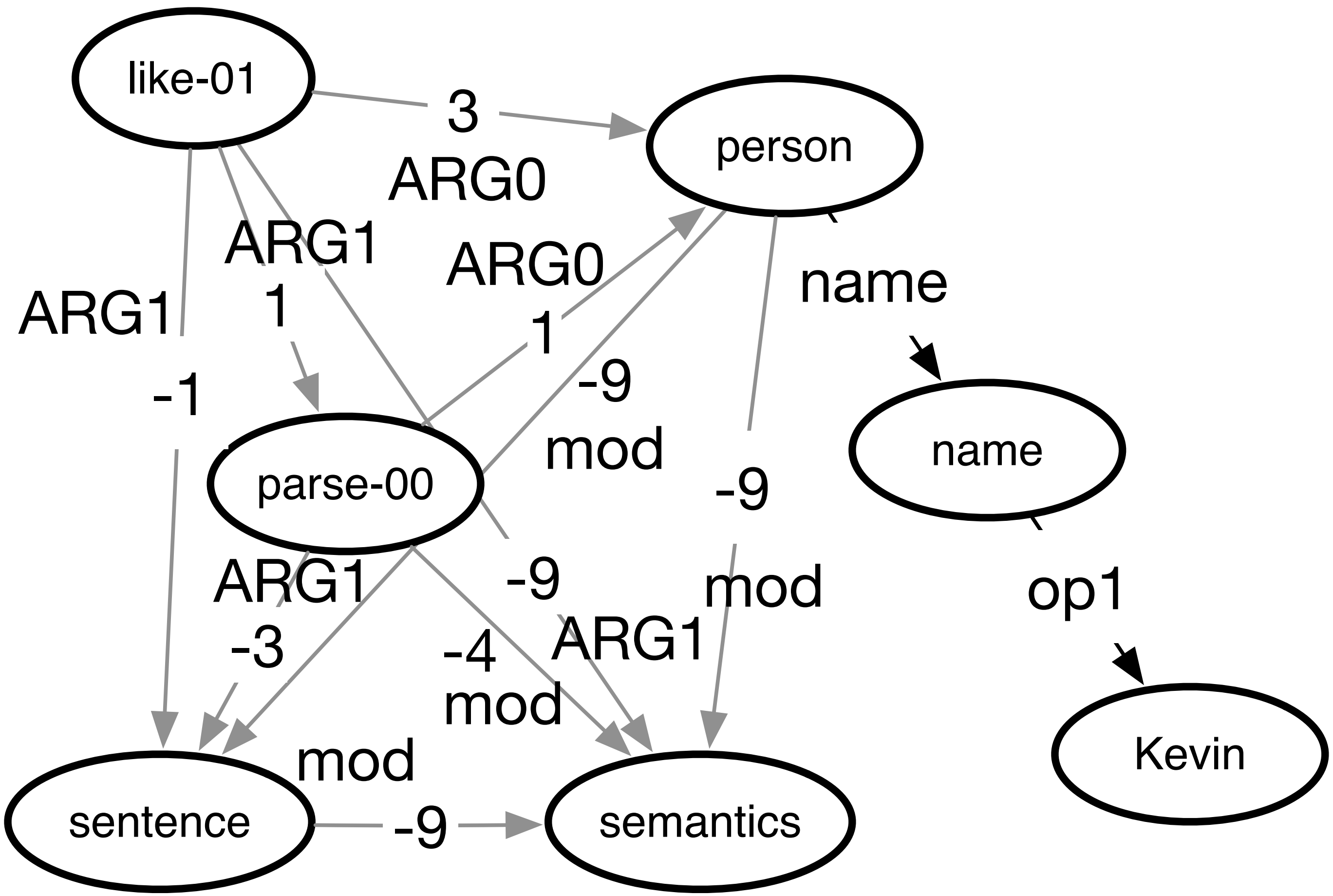
Constraint: Preserving



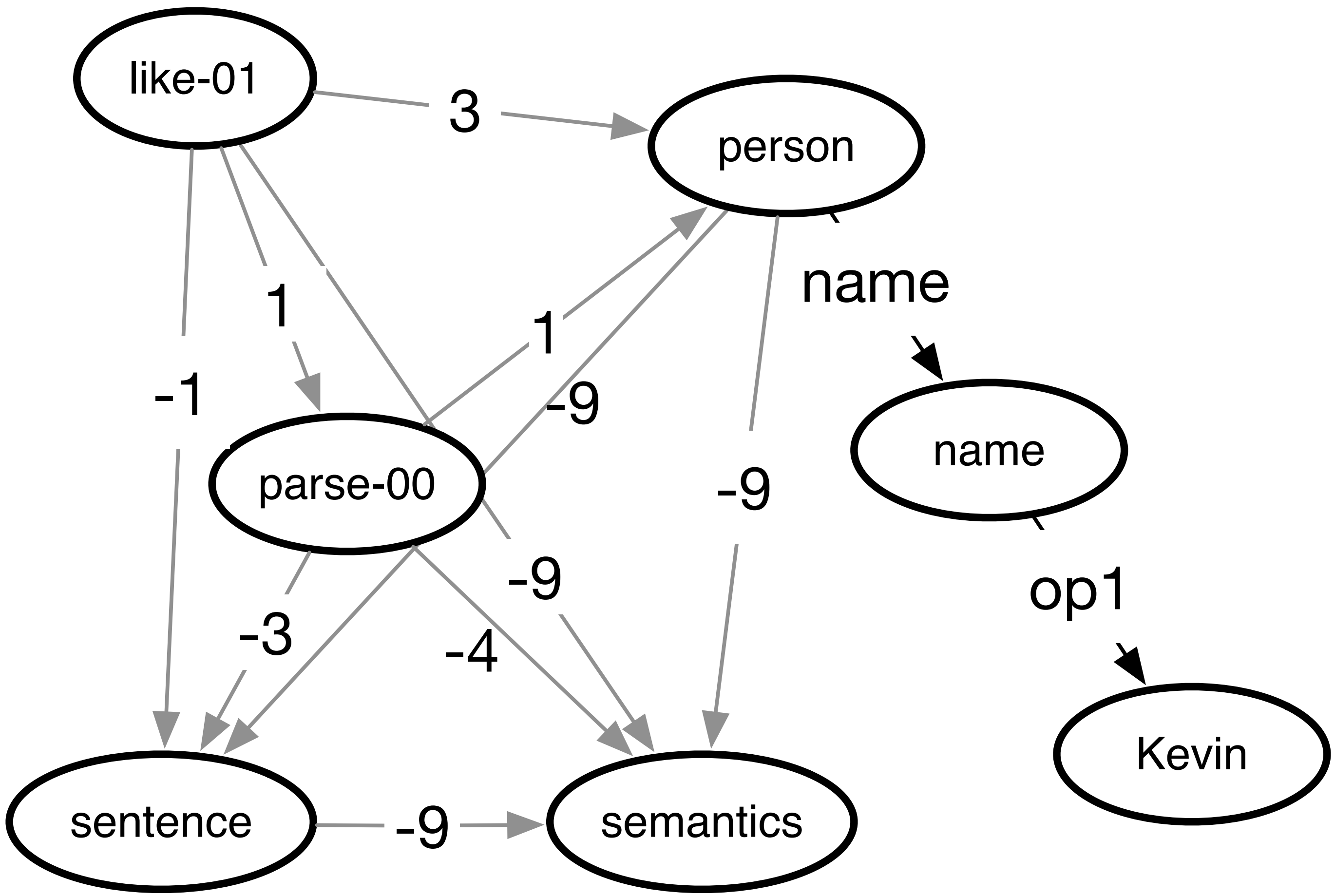
Constraint: Simple



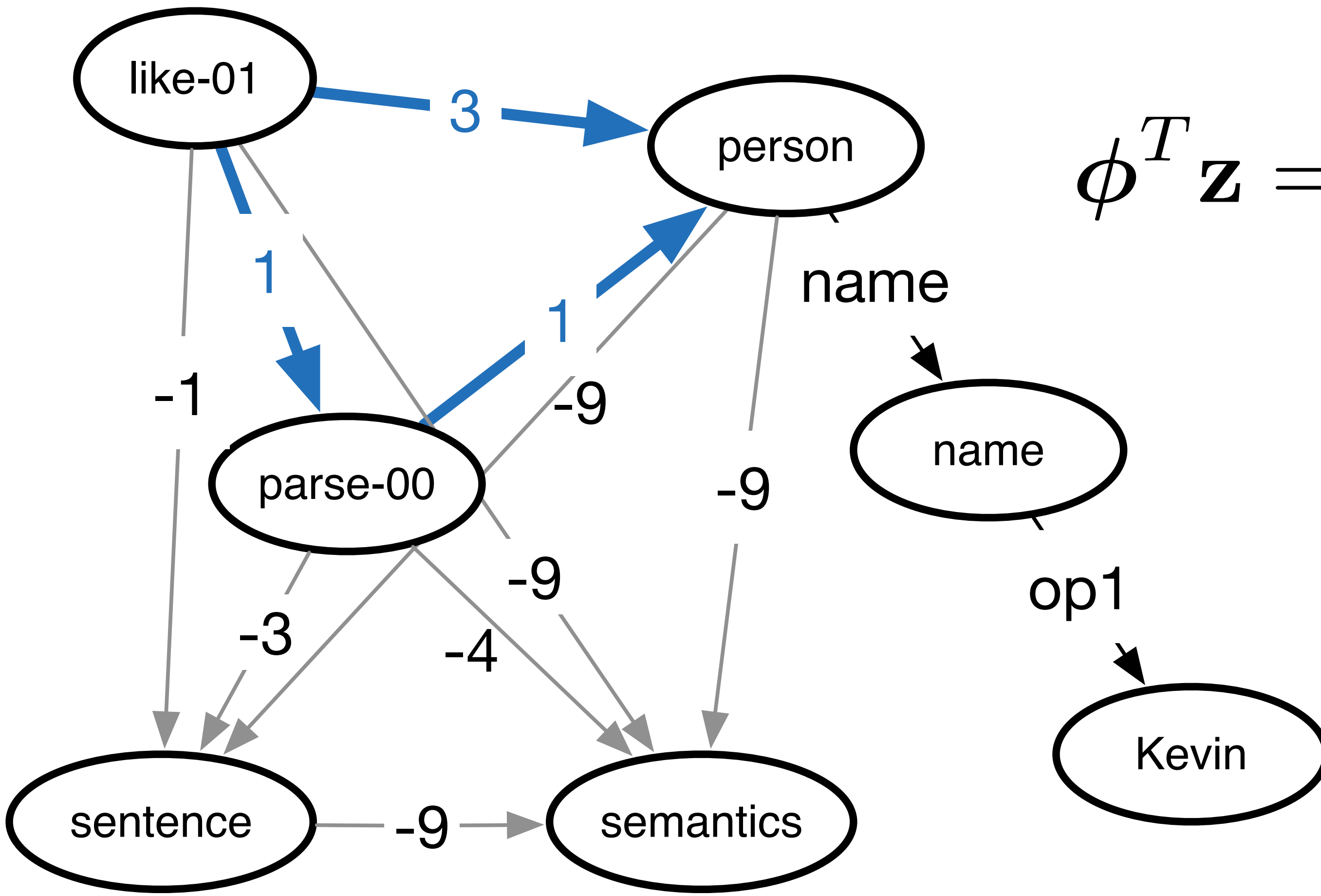
With Weights and Labels Shown



Maximum Weighted Subgraph

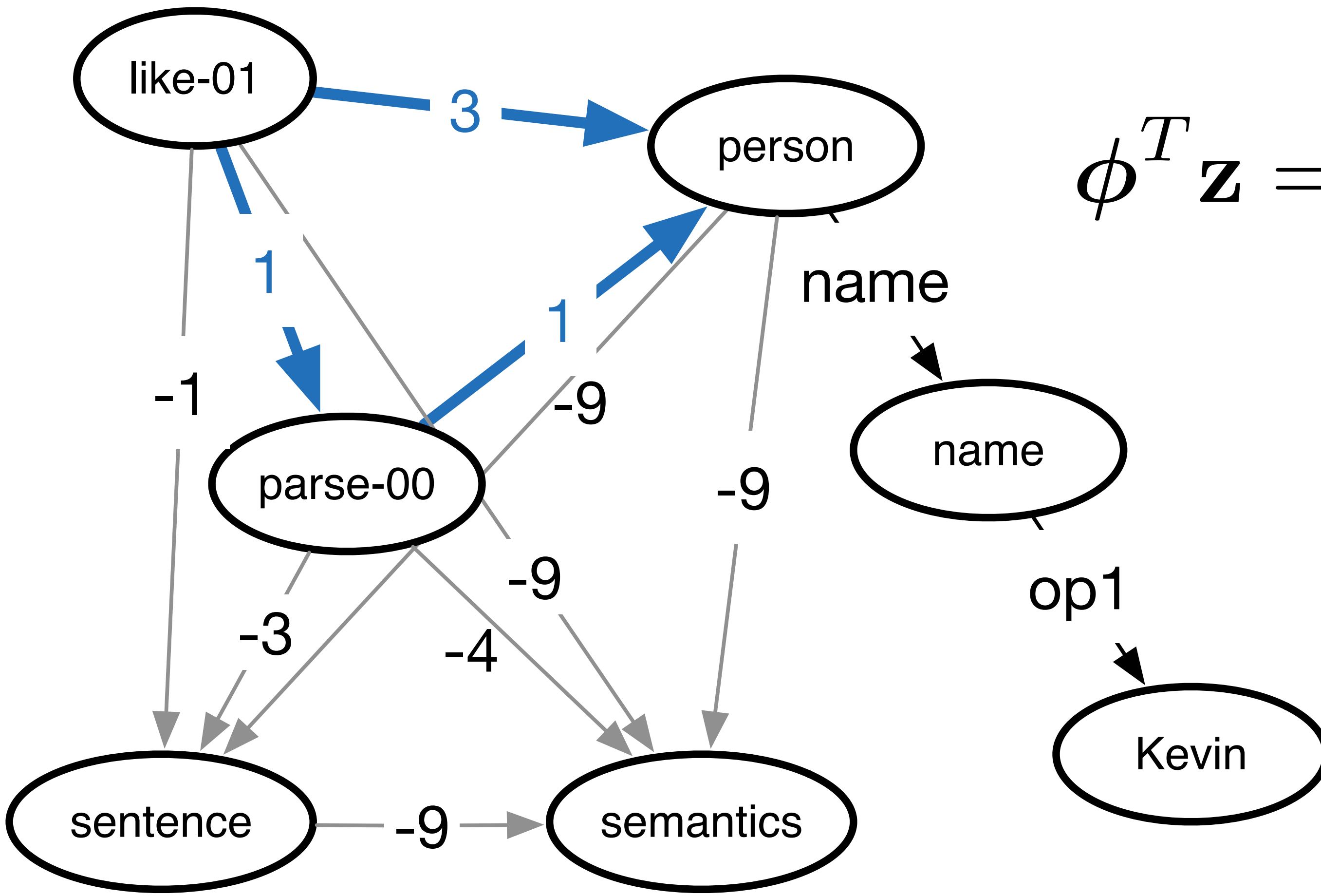


Maximum Weighted Subgraph



$$\phi^T \mathbf{z} = 5$$

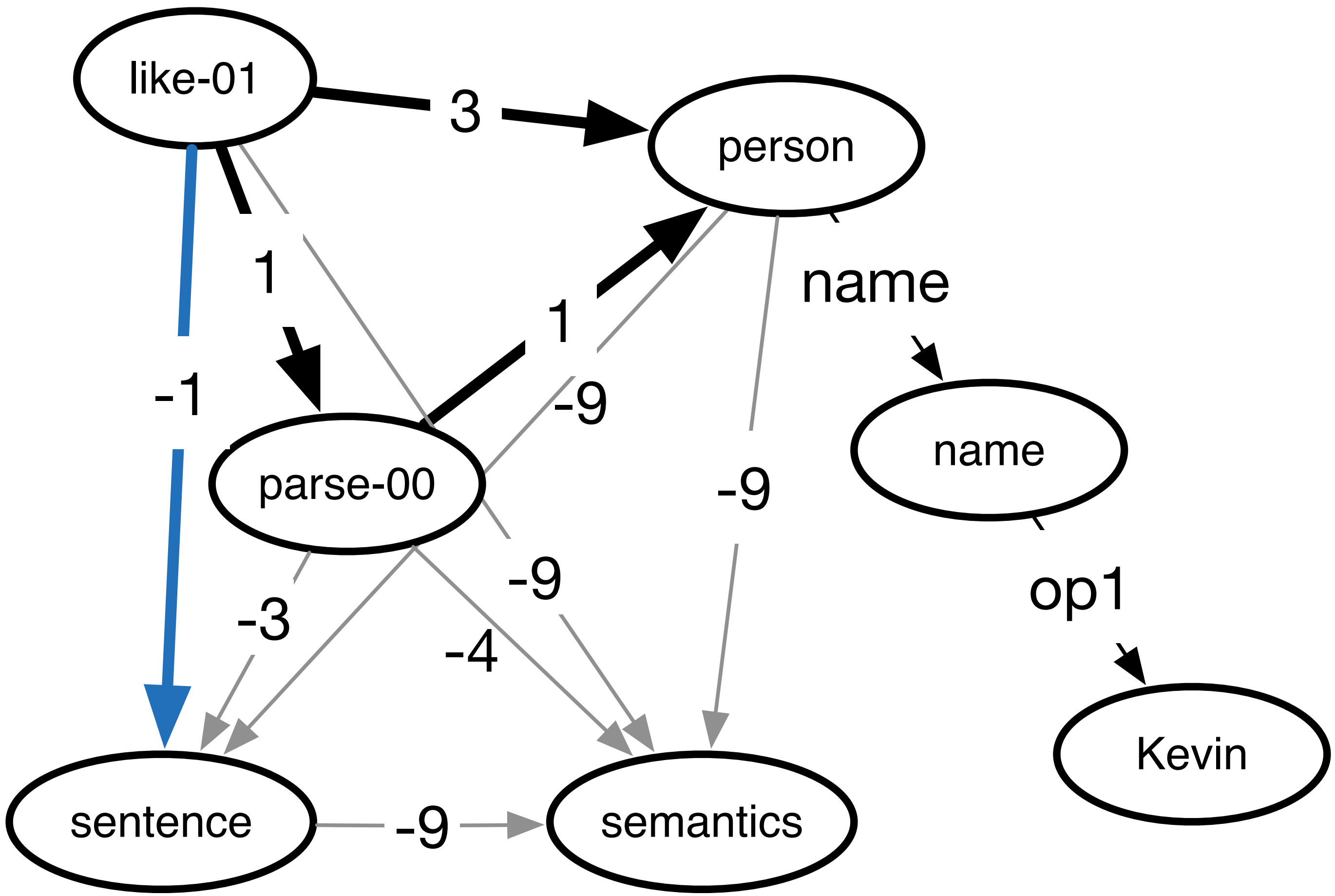
Maximum Weighted Subgraph



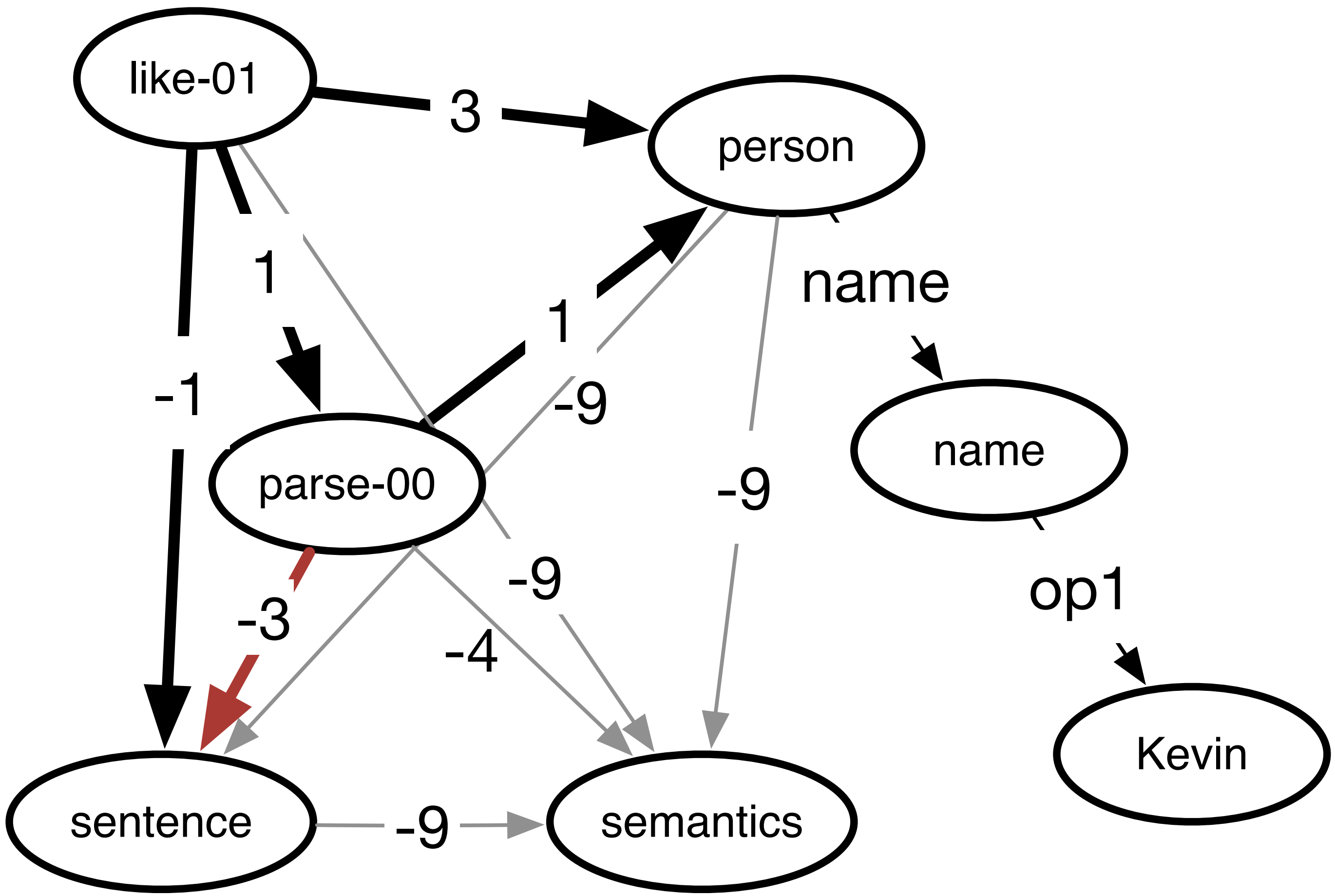
$$\phi^T \mathbf{z} = 5$$

Constraint: Graph must be connected

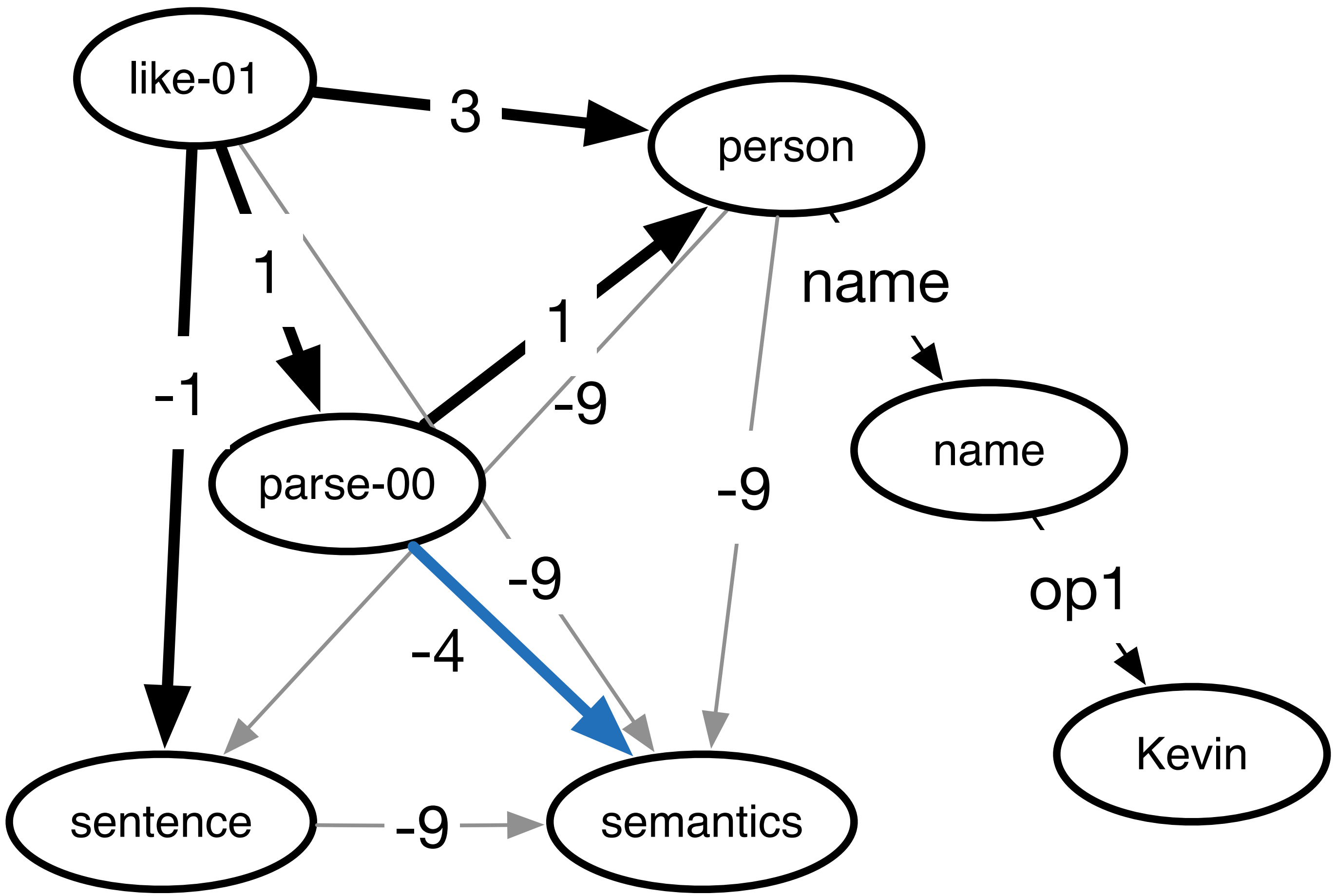
Maximum Spanning, Connected Subgraph (MSCG)



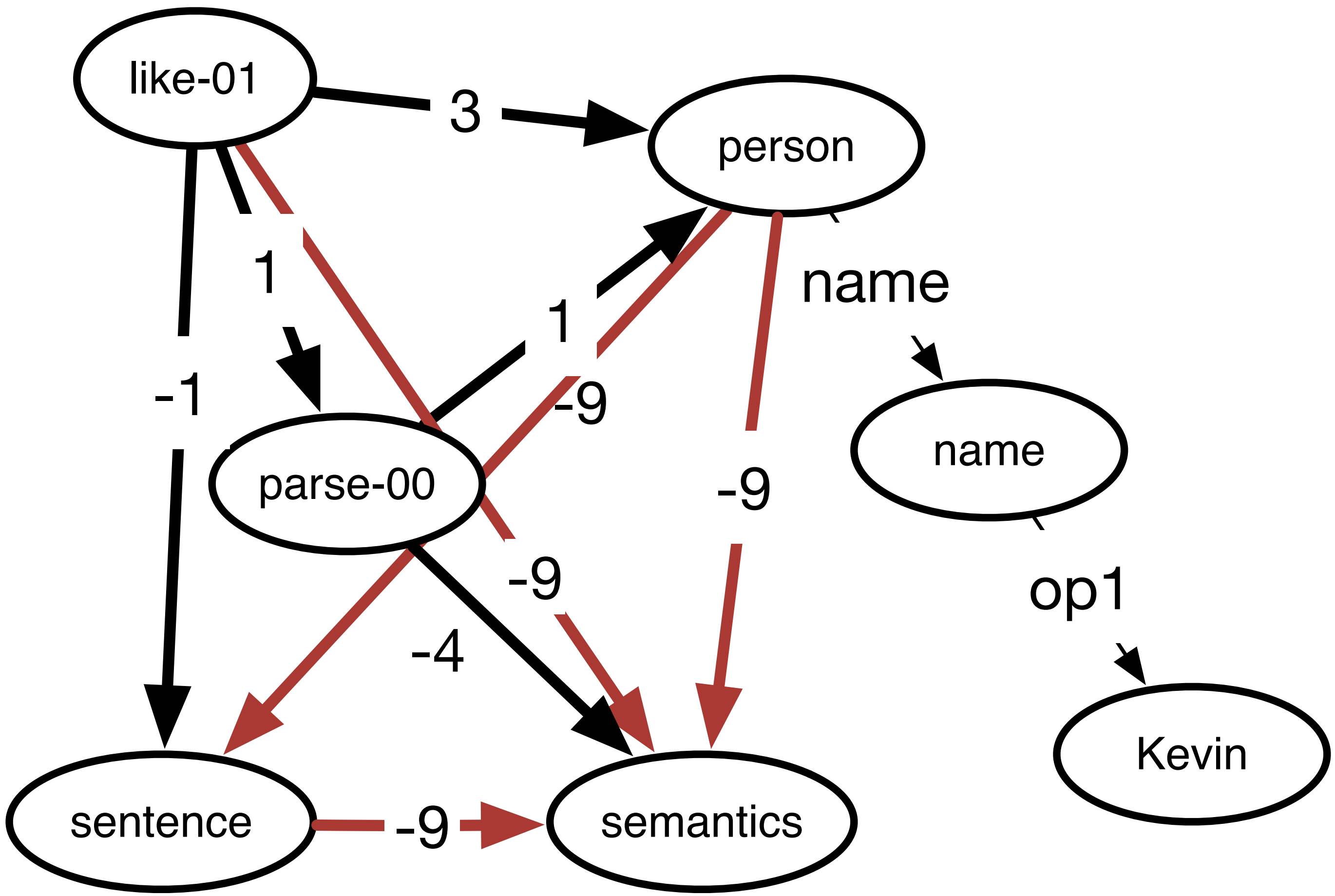
Maximum Spanning, Connected Subgraph (MSCG)



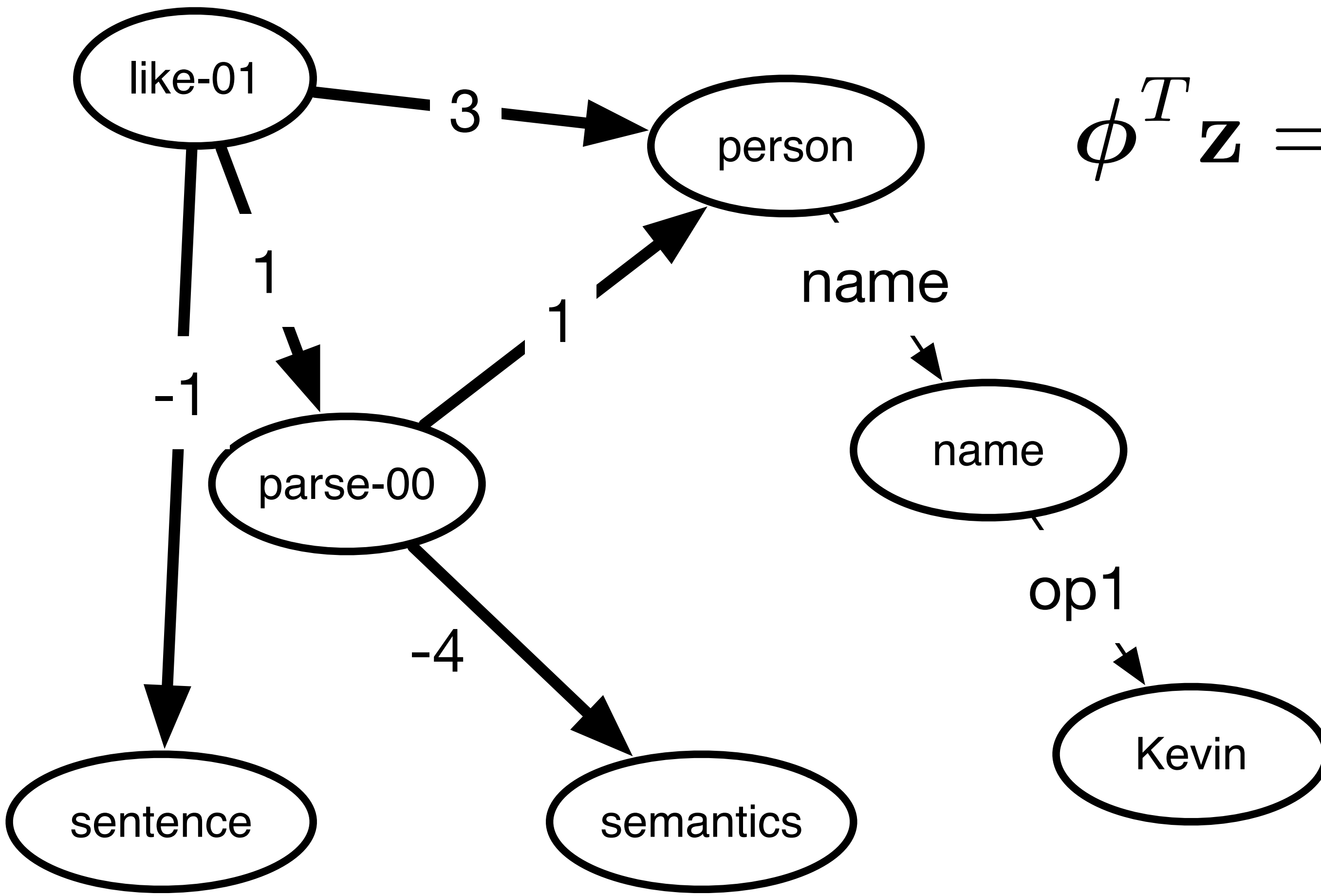
Maximum Spanning, Connected Subgraph (MSCG)



Maximum Spanning, Connected Subgraph (MSCG)

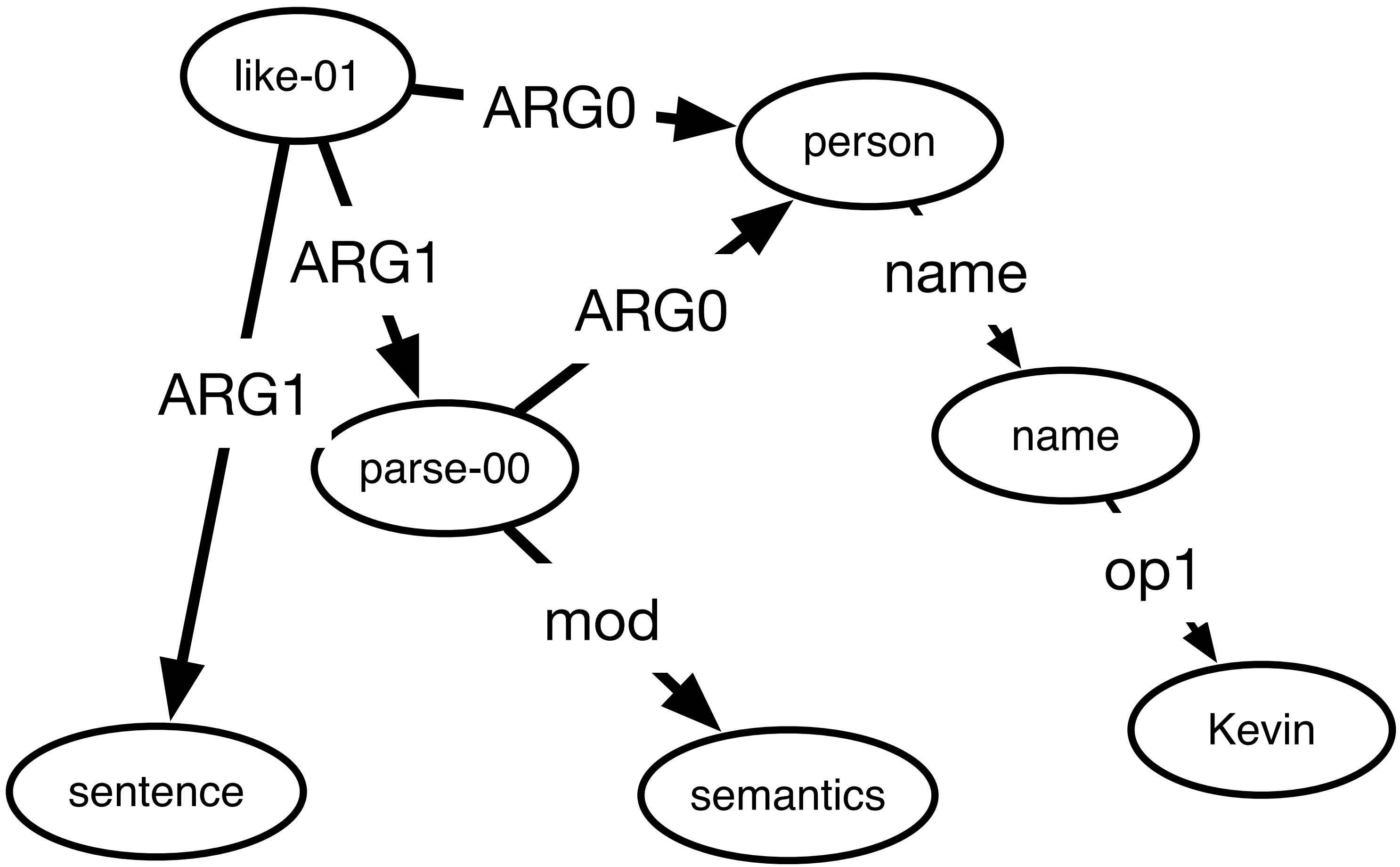


Maximum Spanning, Connected Subgraph (MSCG)



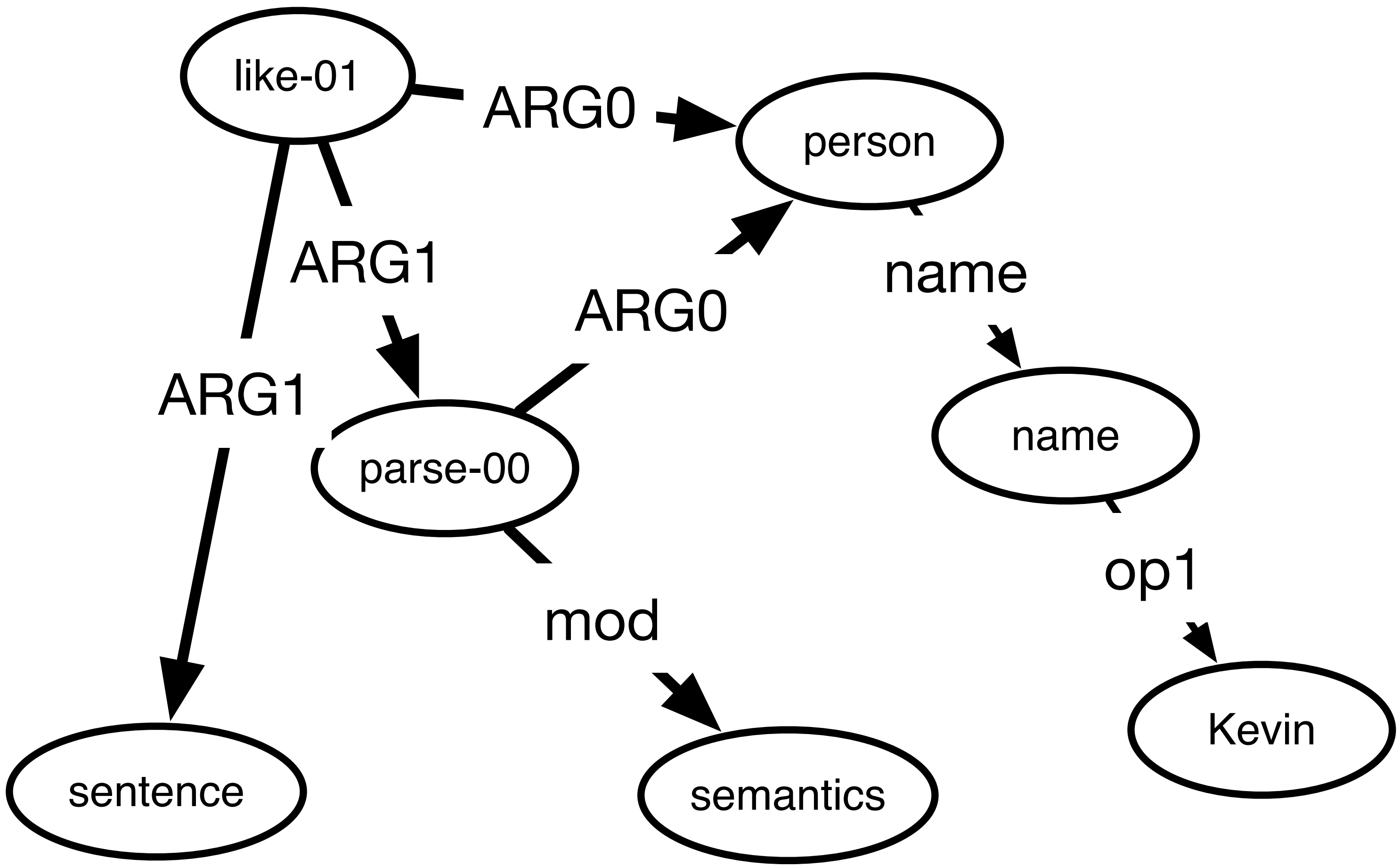
$$\phi^T \mathbf{z} = 0$$

Maximum Spanning, Connected Subgraph (MSCG)



Constraint: Graph must be deterministic

Determinism Constraints



Determinism Constraints

$$\mathbf{z}_{1 \xrightarrow{\text{ARG1}} 2} + \mathbf{z}_{1 \xrightarrow{\text{ARG1}} 3} + \ldots \leq 1$$

$$\mathbf{z}_{2 \xrightarrow{\text{ARG1}} 1} + \mathbf{z}_{2 \xrightarrow{\text{ARG1}} 3} + \ldots \leq 1$$

$$\vdots$$

Determinism Constraints

$$\mathbf{z}_{1 \xrightarrow{\mathrm{ARG1}} 2} + \mathbf{z}_{1 \xrightarrow{\mathrm{ARG1}} 3} + \ldots \leq 1$$

$$\mathbf{z}_{2 \xrightarrow{\mathrm{ARG1}} 1} + \mathbf{z}_{2 \xrightarrow{\mathrm{ARG1}} 3} + \ldots \leq 1$$

$$\vdots$$

$$A\mathbf{z} \leq b$$

Determinism Constraints

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

Preserving, simple, connected, spanning

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Determinism constraints

Solve using Lagrangian relaxation

Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Lagrangian Relaxation Tutorial

$$\boxed{\begin{array}{c} \max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z} \\ \text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b \end{array}}$$

Easy (know how to solve)

Lagrangian Relaxation Tutorial

Easy

$$\boxed{\begin{array}{l} \displaystyle\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z} \\ \text{s.t. } \mathbf{z} \text{ satisfies } A\mathbf{z} \leq b \end{array}}$$

But this makes it hard

Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Easy

Problem #1

Hard

But this makes it hard

Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Easy

Problem #1

Hard

But this makes it hard

$$\lambda^T(\mathbf{b} - A\mathbf{z})$$

Lagrange multipliers ≥ 0

Lagrangian Relaxation Tutorial

Easy

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

But this makes it hard

Problem #1

Hard

$$\max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Add to original objective

Lagrangian Relaxation Tutorial

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Easy

Problem #1

Hard

But this makes it hard

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Minimize over λ

Lagrangian Relaxation Tutorial

Easy

$$\max_{\mathbf{z} \in \mathcal{Z}} \boldsymbol{\phi}^T \mathbf{z}$$

s.t. $\mathbf{z}$ satisfies $A\mathbf{z} \leq b$

Problem #1

Hard

But this makes it hard

$$\approx$$

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

Problem #2

Easy

Not always equivalent, as we shall see

Solving Problem #2

- Problem #2 (aka “Lagrange Dual”):

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

- For a given λ, the max can be solved using algorithm given before (preprocessing + MSCG)
- To minimize over lambda
  – Use subgradient descent

Solving Problem #2

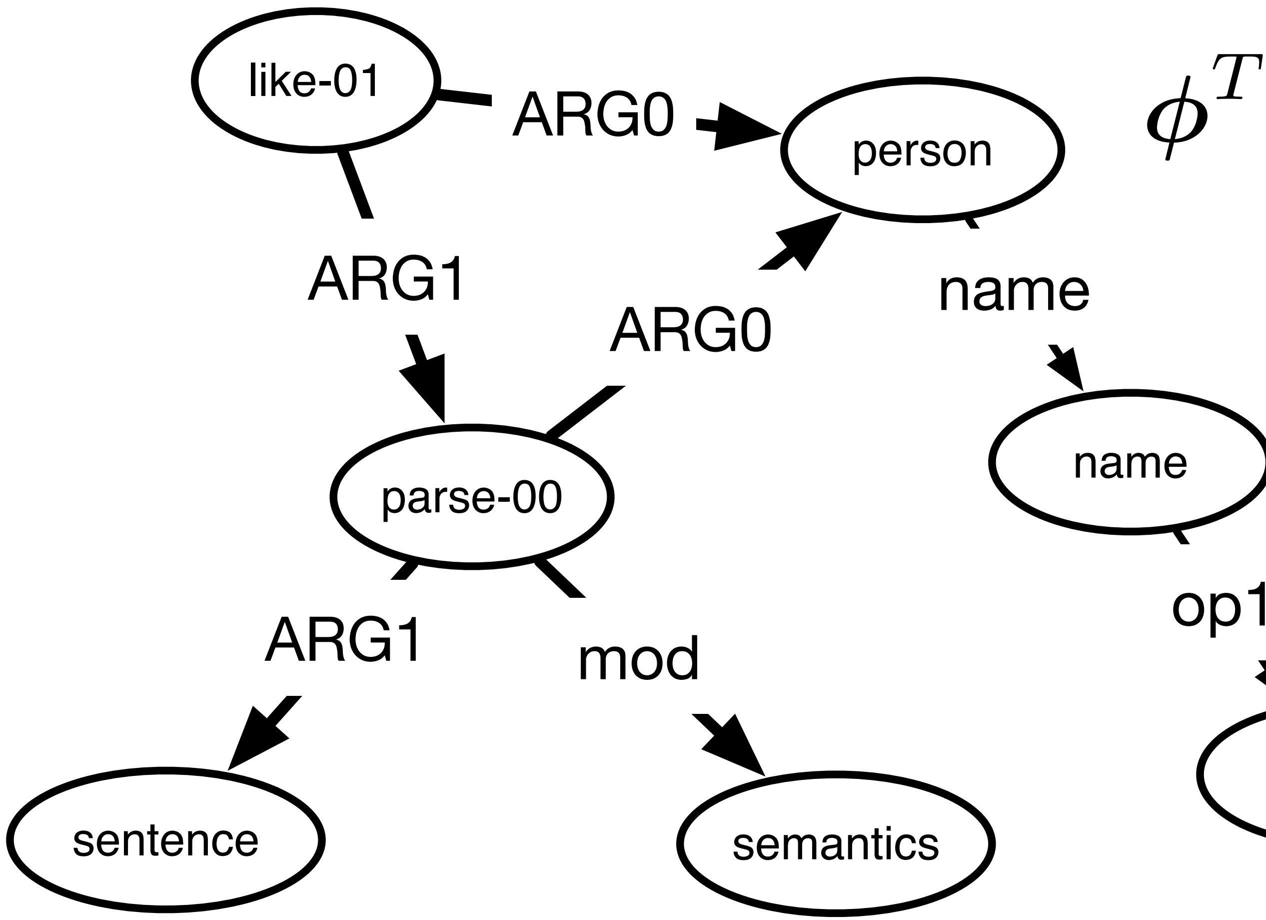
- Problem #2 (aka "Lagrange Dual"):

$$\min_{\lambda \geq 0} \max_{\mathbf{z} \in \mathcal{Z}} \phi^T \mathbf{z} + \lambda^T (\mathbf{b} - A\mathbf{z})$$

- For a given λ, the max can be solved using algorithm given before (preprocessing + MSCG)
- To minimize over lambda
  - Use subgradient descent

If constraints are not satisfied at minimum, then Problem #1 ≠ Problem #2

After subgradient descent



$$\phi^T \mathbf{z} = -2$$

Summary: Output Graph Properties

- Maximum weight
- Preserving
- Simple
- Spanning (all nodes)
- Connected
- Deterministic

Features & Training

- Features
  - Edge bias
  - Edge label
  - Head concept, tail concept, head word, tail word
  - Dependency path (dependency edge labels and POS on the shortest path between any two words in the span)
  - Various distance features
  - Within fragment edge indicator
  - Various conjunctions of above features
- Weights trained using AdaGrad structured perceptron

Experiments

- Alignment
- Parsing
  - Graph-based parsing
    - Concept identification
    - Relation identification
      - Maximum spanning connected graph algorithm (MSCG)
      - Graph determinism constraints using Lagrangian relaxation
    - **Experiments**
  - Transition-based parsing
  - Parsing using syntax-based MT
- Evaluation
- Graph Grammars and Automata
- Applications

Experiments

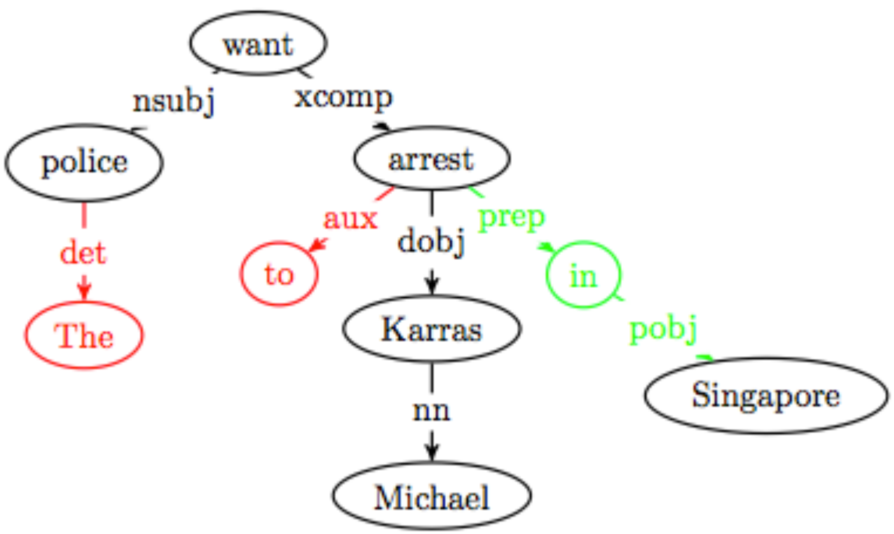
	ACL 2014	Now
Full System (gold concepts)	80% Smatch	81% Smatch
Full System	58% Smatch	62% Smatch

- Data: LDC2013E117
  - 4,000 training instances
  - 2,000 test
  - 2,000 dev

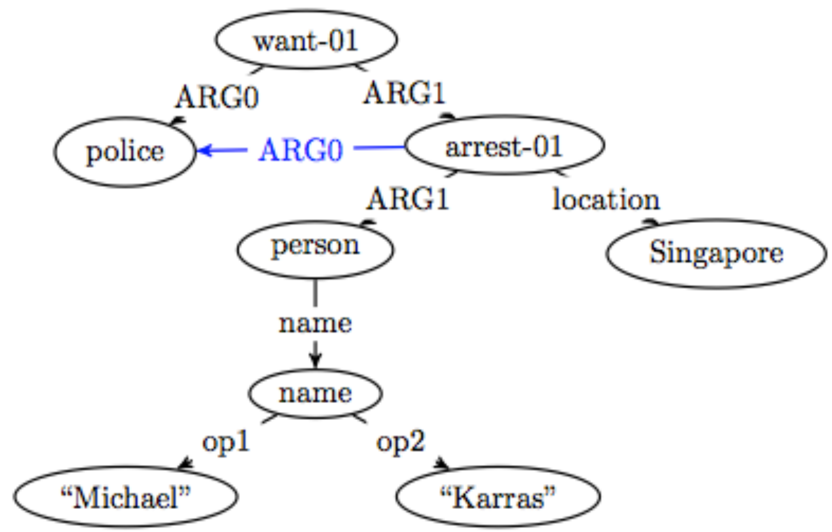
Transition-based AMR Parsing (Wang et al, NAACL 2015)

- Convert dependency tree into AMR graph
- Motivation: only a few difference between syntactic dependencies and AMR

Dependency tree



AMR graph

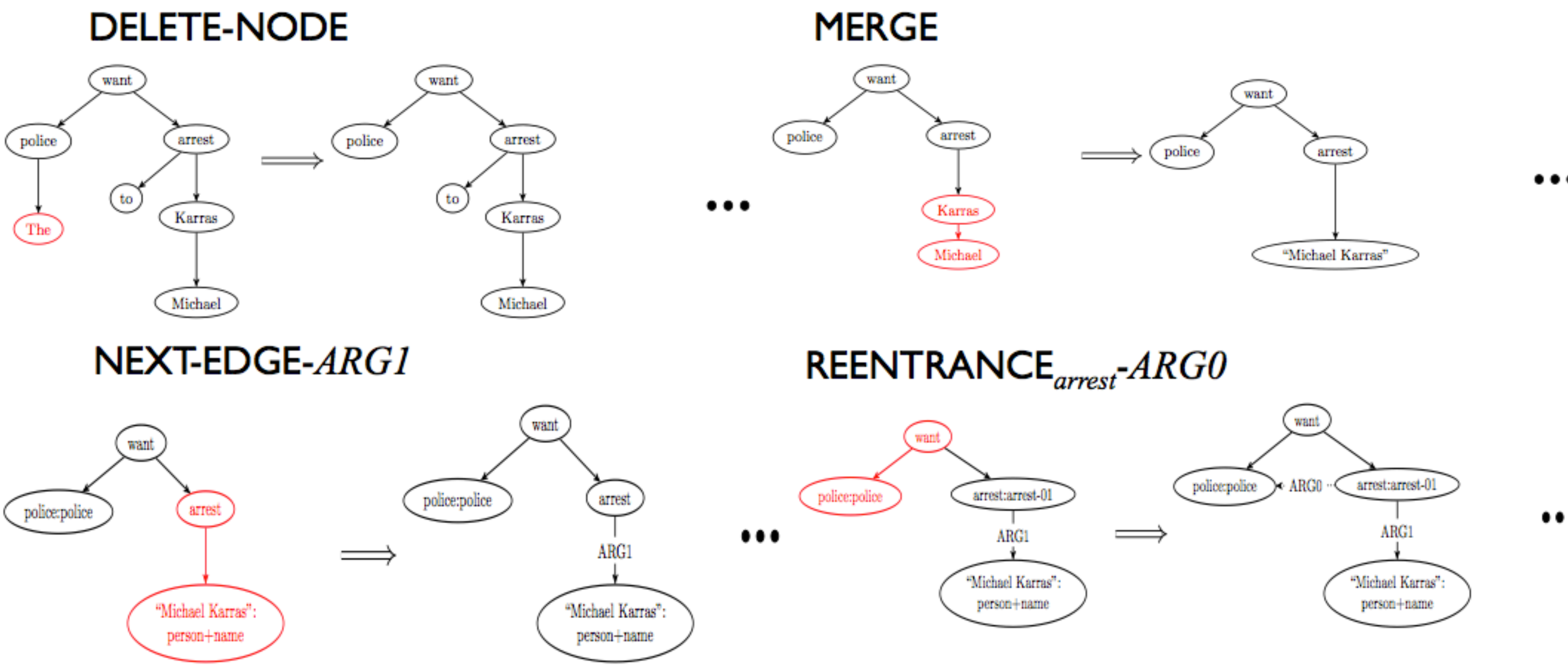


(graphic thanks to Chuan Wang)

Transition-based AMR Parsing

- Actions applied to graph in post-order traversal
- Parser actions
  - **NEXT-EDGE-l_r** (attach edge and move to next word)
  - **SWAP-l_r** (swap nodes and attach with edge)
  - **REATTACH$_k$-l_r** (delete edge and reattach to already processed node)
  - **REPLACE-HEAD** (replace node with another node)
  - **REENTRANCE$_k$-l_r** (attach edge to already processed node)
  - **MERGE** (merge two nodes)
  - **NEXT-NODE-l_c** (label with concept and move to next word)
  - **DELETE-NODE** (deletes a word)

Transition-based AMR Parsing



(graphic thanks to Chuan Wang)

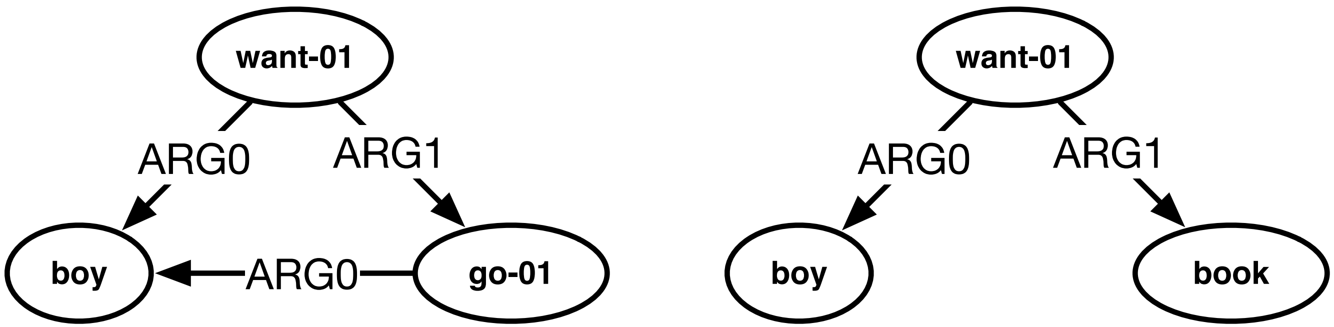
AMR Parsing using Syntax-based MT (Pust et al, 2015)

- Idea: already have sophisticated string-to-tree syntactic MT systems. Use them for AMR parsing
- Approach: convert AMR graphs into trees suitable for training string-to-tree MT systems
- Important features:
  - Language model on the linearized AMR
  - Semantic categories built using WordNet
- Large performance gains JAMR (7 smatch points)

Evaluation

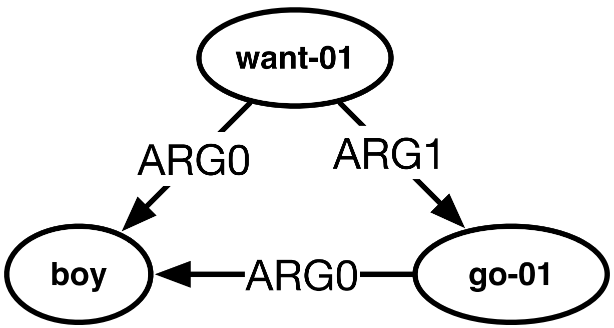
- Alignment
- Parsing
- **Evaluation**
- Graph Grammars and Automata
- Applications

Evaluation: Smatch (Cai & Knight, 2013)

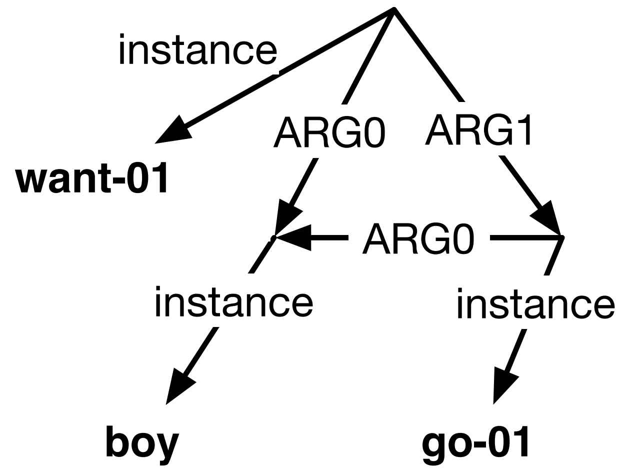


Want a number which indicates the similarity between two graphs

Evaluation: Smatch

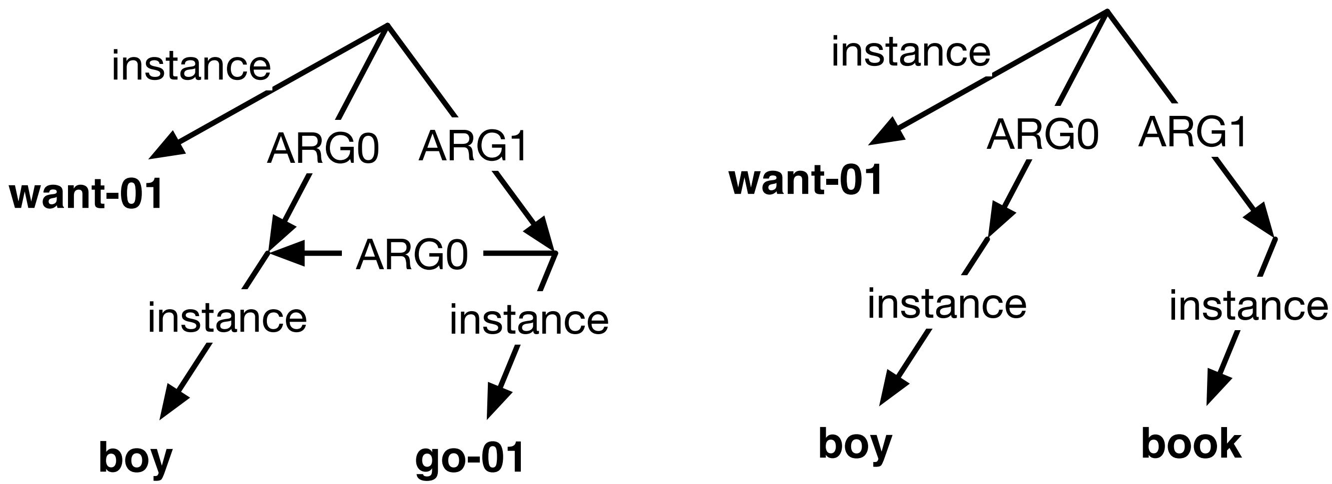



```
(a / want-01
     :ARG0 (b / boy
     :ARG1 (c / go-01
          :ARG0 b))
```

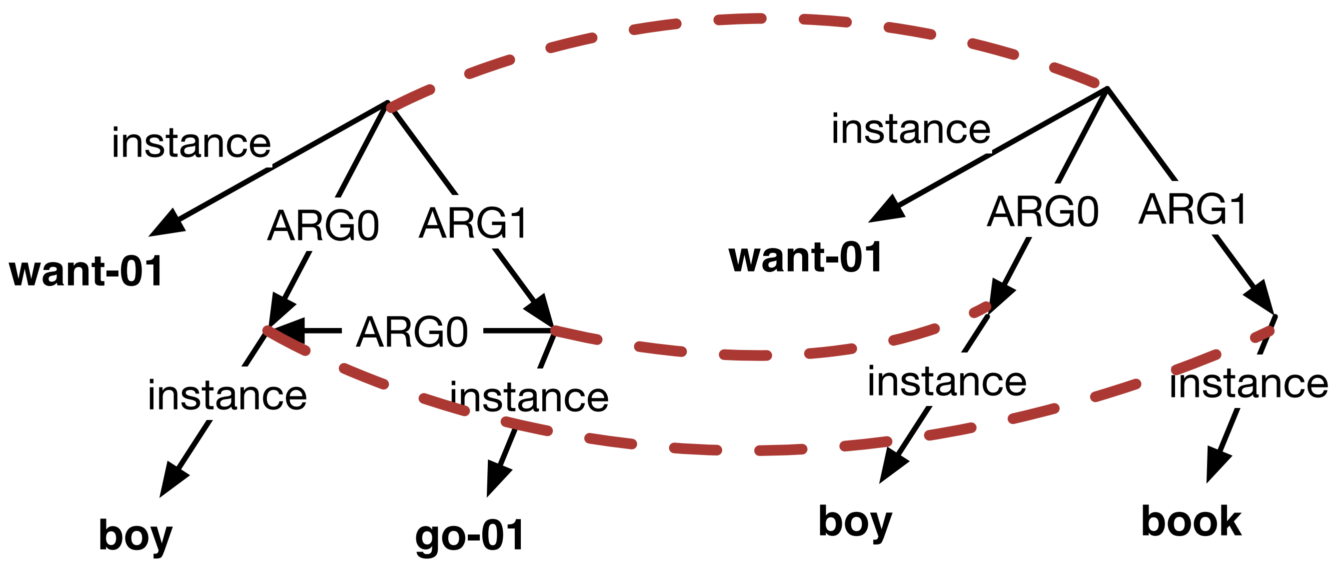


AMR graph w/ explicit instance edges

Evaluation: Smatch

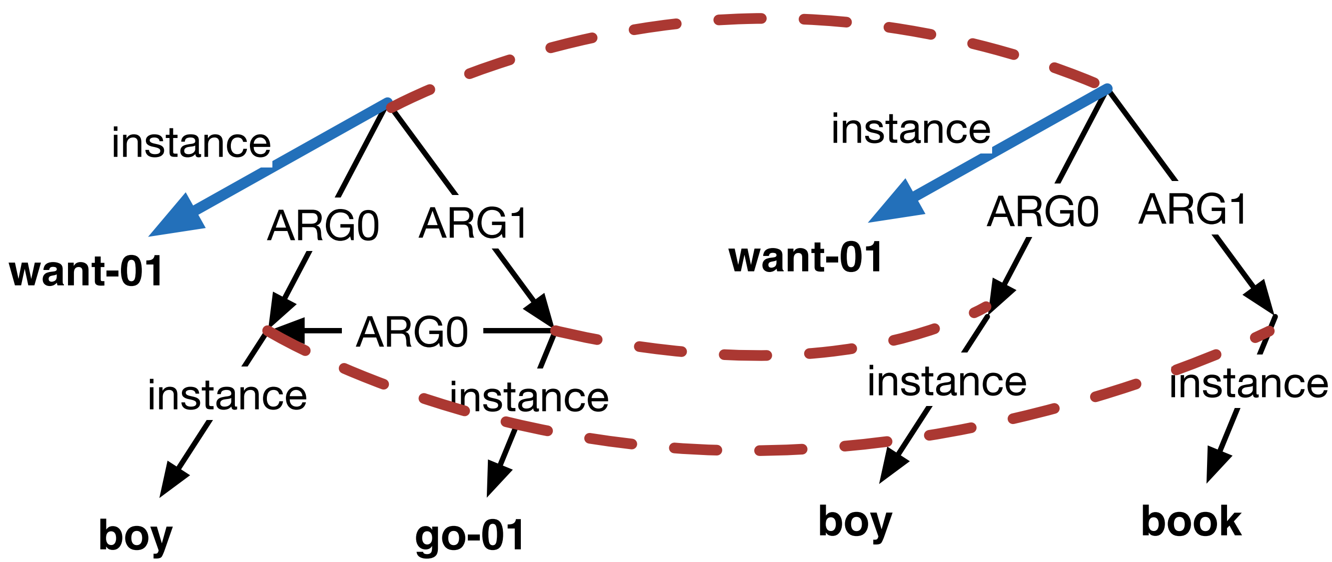


Evaluation: Smatch



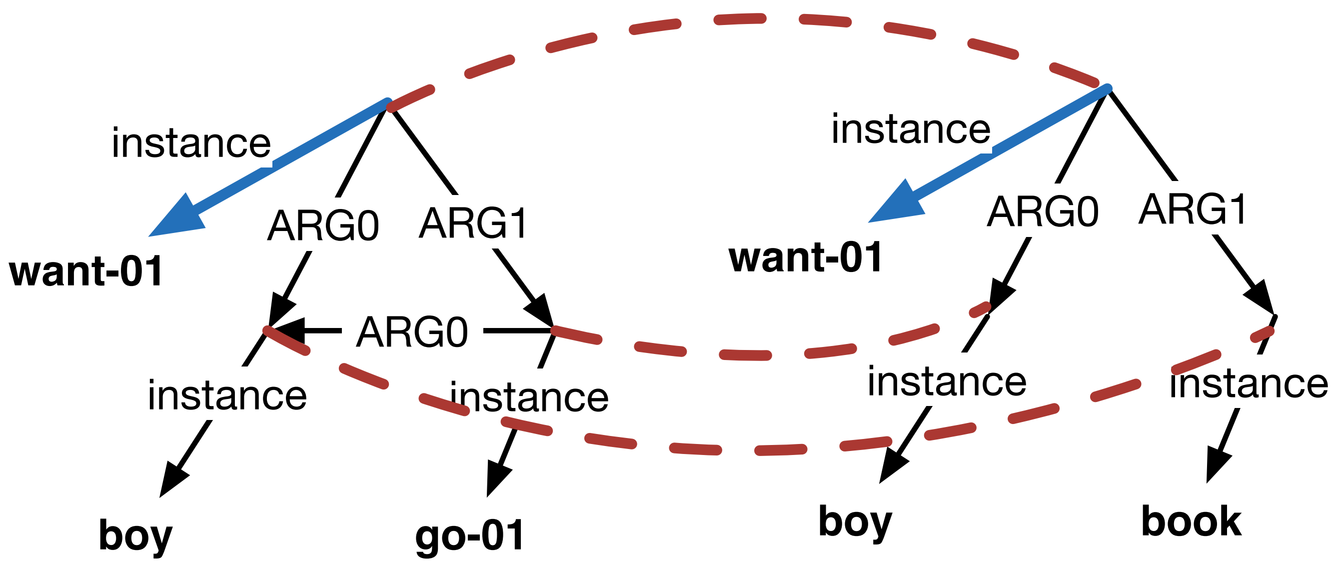
Consider an alignment between the nodes

Evaluation: Smatch



$$\text{f-score} = F_1 \text{ of identical matching edges}$$
$$= 2\ \text{Match}/(\text{Total}_1 + \text{Total}_2)$$
$$= 2 / (6 + 5) = .18$$

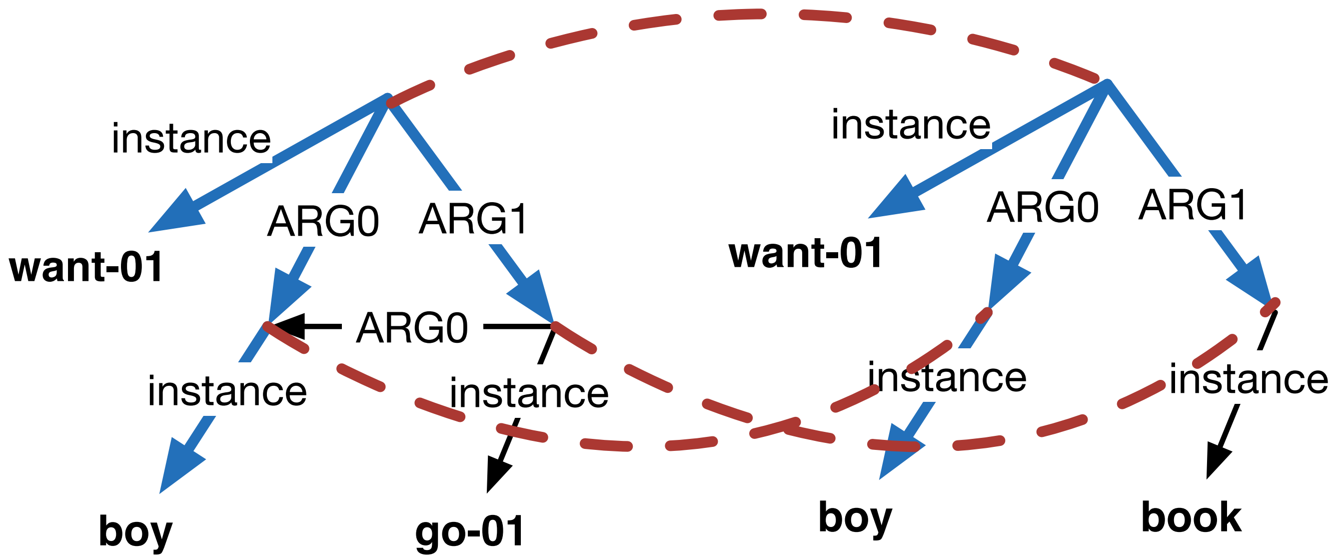
Evaluation: Smatch



Smatch score = maximum f-score over all possible alignments

NP hard => approximate inference to find highest scoring alignment

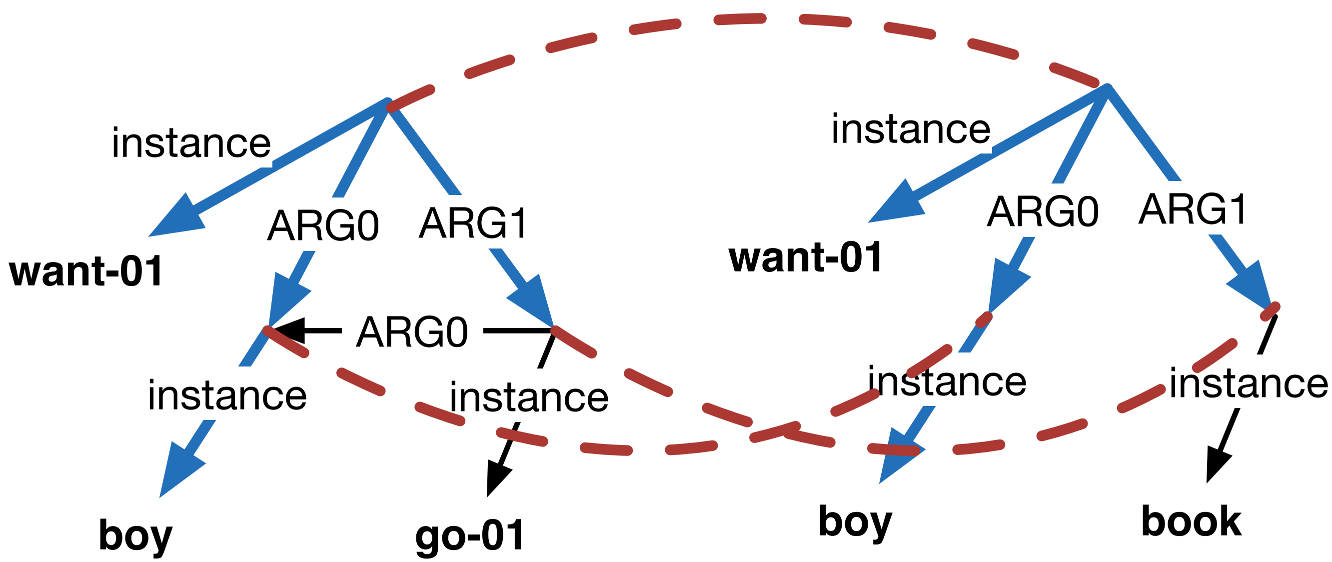
Evaluation: Smatch



Smatch score = 8 / (6 + 5) = .73

Highest scoring alignment

Evaluation: Smatch



Multi-lingual version of Smatch: AMRICA demo by Naomi Saphra at NAACL 2015

Roadmap

- Alignment
- Parsing
- Evaluation
- **Graph Grammars and Automata**
  - Background: CFGs and tree substitution grammars
  - Hyperedge Replacement Grammars (HRGs)
  - Directed Acyclic Graph (DAG) Automata
- Applications

Motivation for Graph Grammars

- String and tree grammars, automata, transducers, etc widely used in NLP applications
  - Phrase structure parsers, syntactic MT systems
- Semantics (like AMR) is represented as graphs

We would like grammars, automata, transducers, etc over graphs

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Context Free Grammar (CFG)

<u>Grammar</u>

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

<u>Example derivation</u>

S

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Example derivation

S

$\Rightarrow_1$ NP VP

Context Free Grammar (CFG)

<u>Grammar</u>

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

<u>Example derivation</u>

S

$\Rightarrow_1$ NP VP

$\Rightarrow_3$ NP V NP

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Example derivation

S

$\Rightarrow_1$ NP VP

$\Rightarrow_3$ NP V NP

$\Rightarrow_4$ NP like NP

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Example derivation

S

$\Rightarrow_1$ NP VP

$\Rightarrow_3$ NP V NP

$\Rightarrow_4$ NP like NP

$\Rightarrow_6$ NP like ice cream

Context Free Grammar (CFG)

<u>Grammar</u>

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

<u>Example derivation</u>

S

$\Rightarrow_1$ NP VP

$\Rightarrow_3$ NP V NP

$\Rightarrow_4$ NP like NP

$\Rightarrow_6$ NP like ice cream

$\Rightarrow_2$ We like ice cream

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Derivation

Example derivation

$$
\left\{
\begin{array}{l}
\quad \text{S} \\
\Rightarrow_1 \text{NP VP} \\
\Rightarrow_3 \text{NP V NP} \\
\Rightarrow_4 \text{NP like NP} \\
\Rightarrow_6 \text{NP like ice cream} \\
\Rightarrow_2 \text{We like ice cream}
\end{array}
\right.
$$

Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Example derivation

Derivation

$$\begin{aligned} & S \\ \Rightarrow_1\ & NP\ VP \\ \Rightarrow_3\ & NP\ V\ NP \\ \Rightarrow_4\ & NP\ \text{like}\ NP \\ \Rightarrow_6\ & NP\ \text{like ice cream} \\ \Rightarrow_2\ & \underbrace{\text{We like ice cream}}_{\text{Yield} = \textbf{string}} \end{aligned}$$

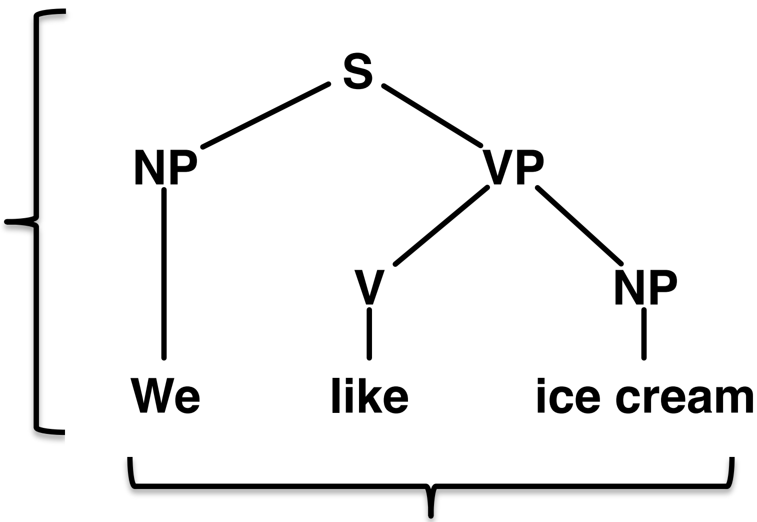
Context Free Grammar (CFG)

Grammar

1) S -> NP VP
2) NP -> We
3) VP -> V NP
4) V -> want
5) V -> like
6) NP -> ice cream

Example derivation

Derivation tree

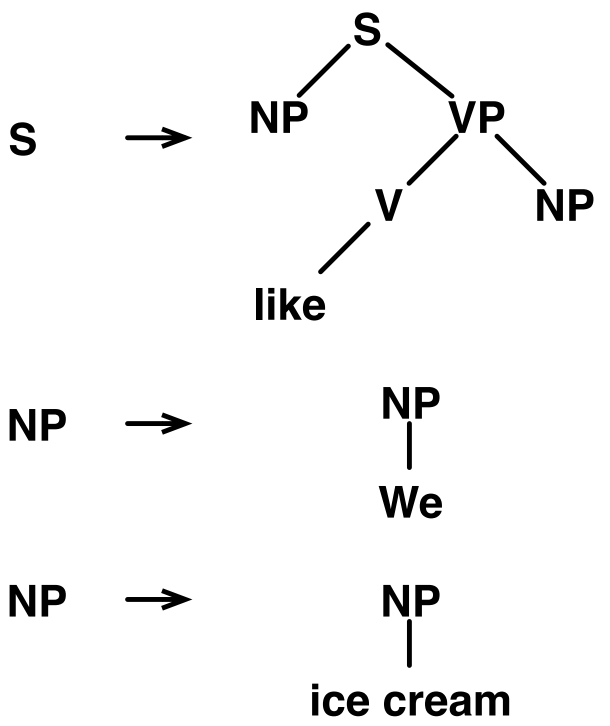


Yield = **string**

Language over strings (yield), and trees (derivations)

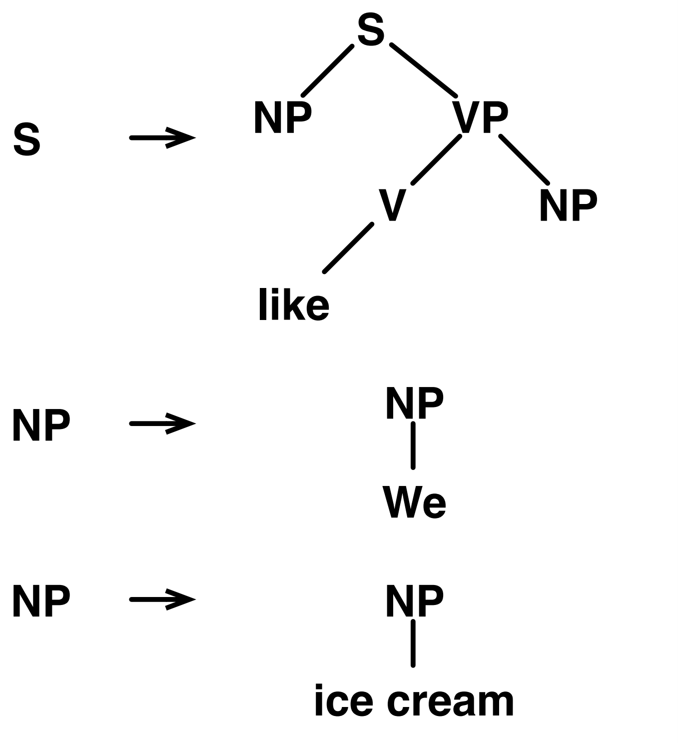
Tree Substitution Grammar (TSG)

Grammar



Tree Substitution Grammar (TSG)

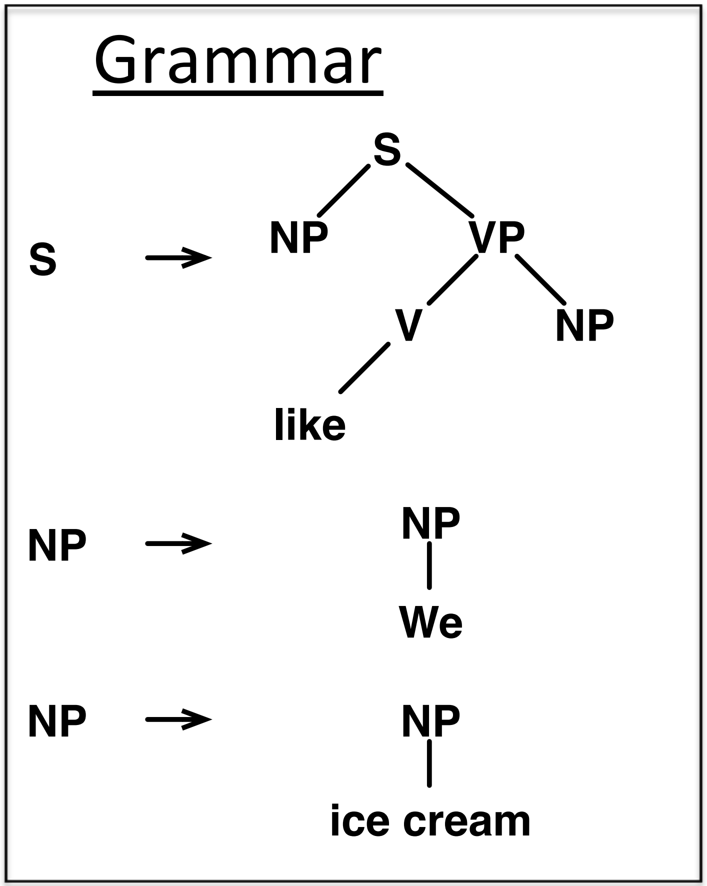
Grammar

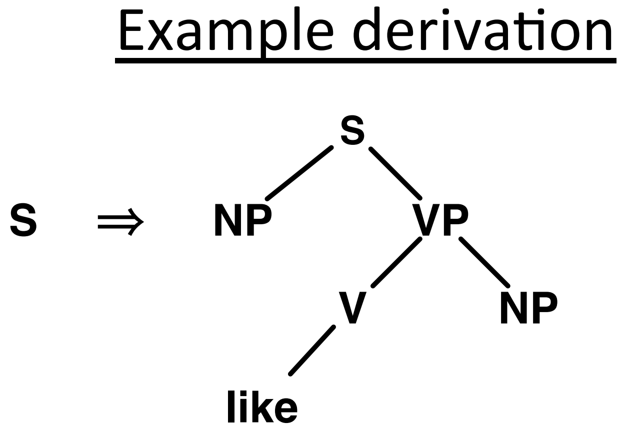


Example derivation

S

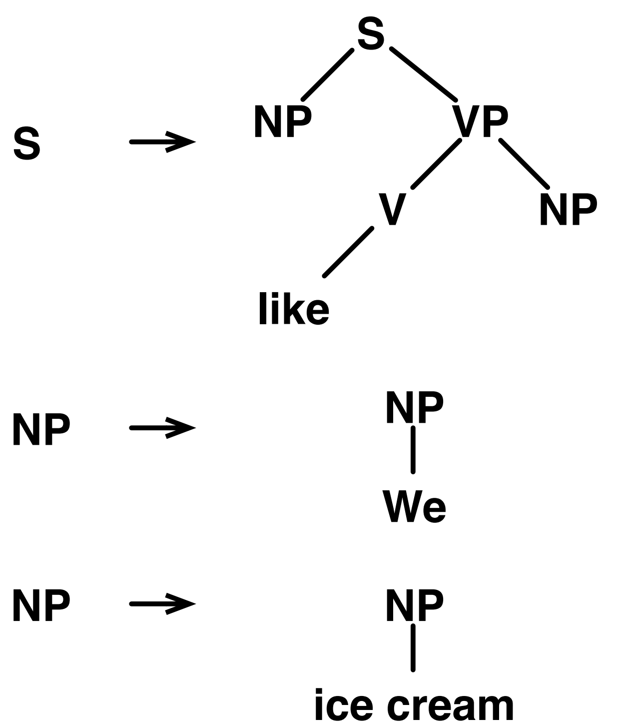
Tree Substitution Grammar (TSG)



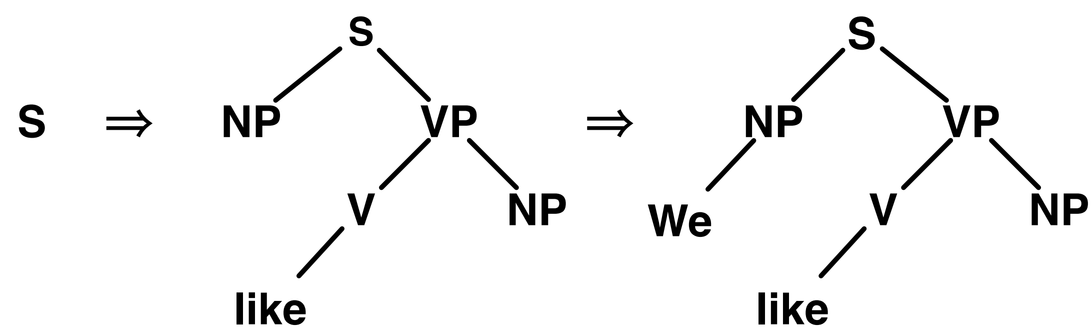


Tree Substitution Grammar (TSG)

Grammar

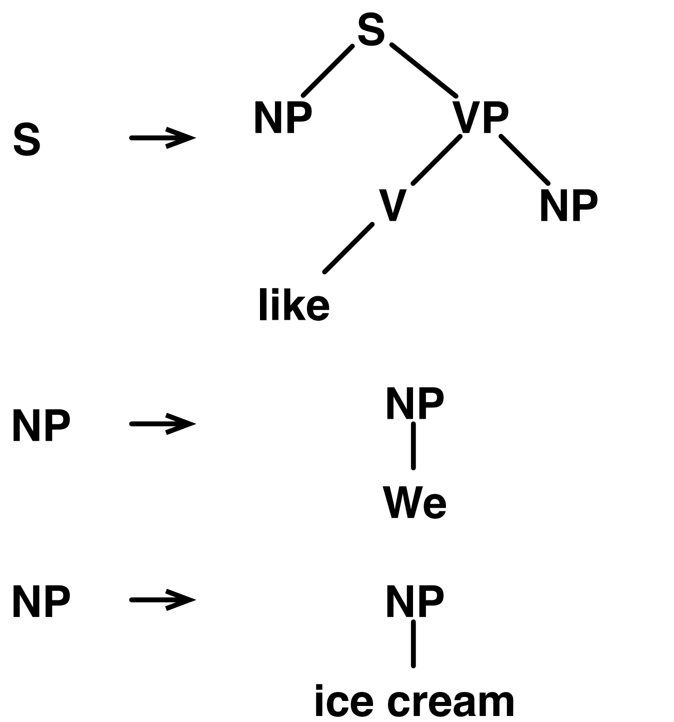


Example derivation

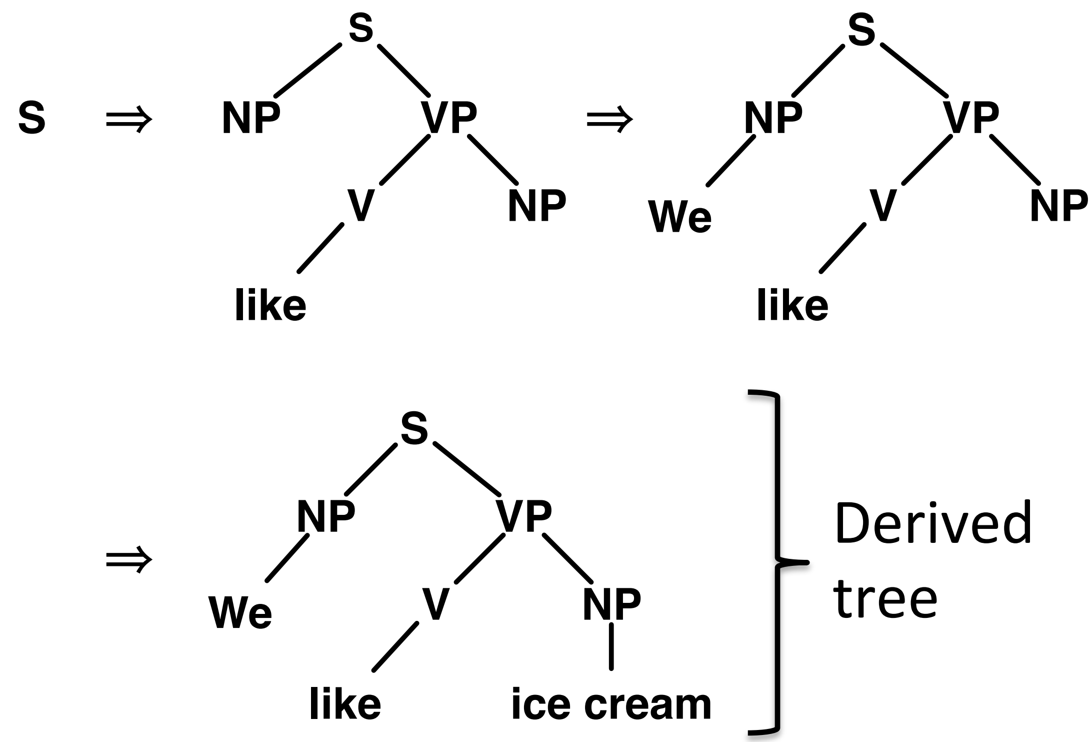


Tree Substitution Grammar (TSG)

Grammar

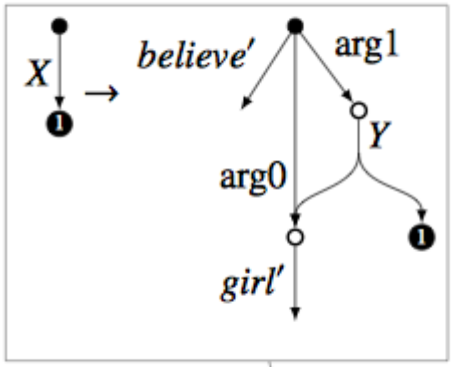


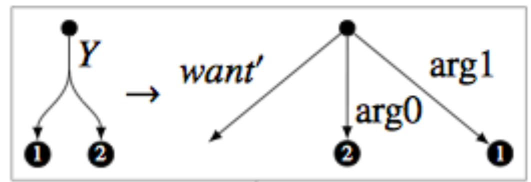
Example derivation



Hyperedge Replacement Grammar (HRG)

Grammar rules (some of them)

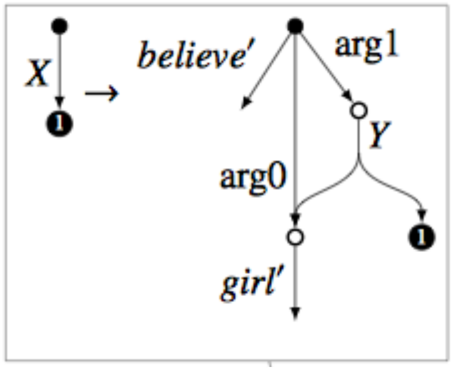


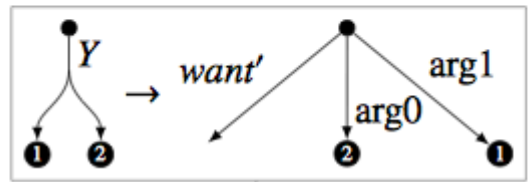


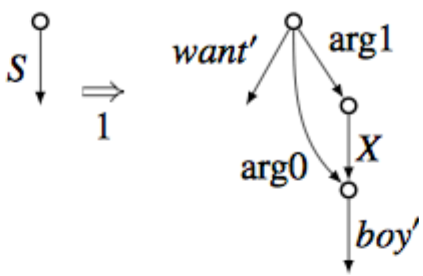
(figure from Chiang et al, 2013)

Hyperedge Replacement Grammar (HRG)

Grammar rules (some of them)



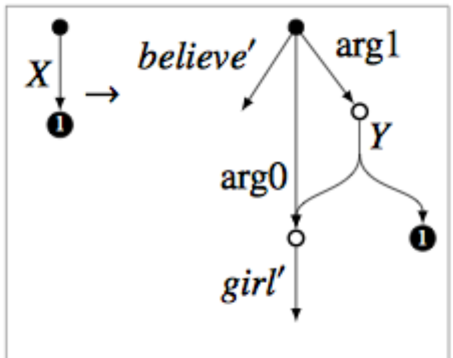


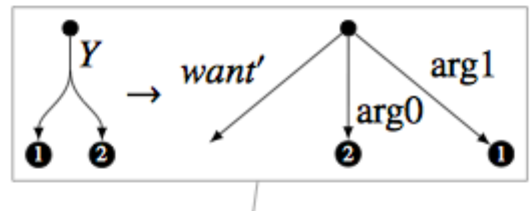


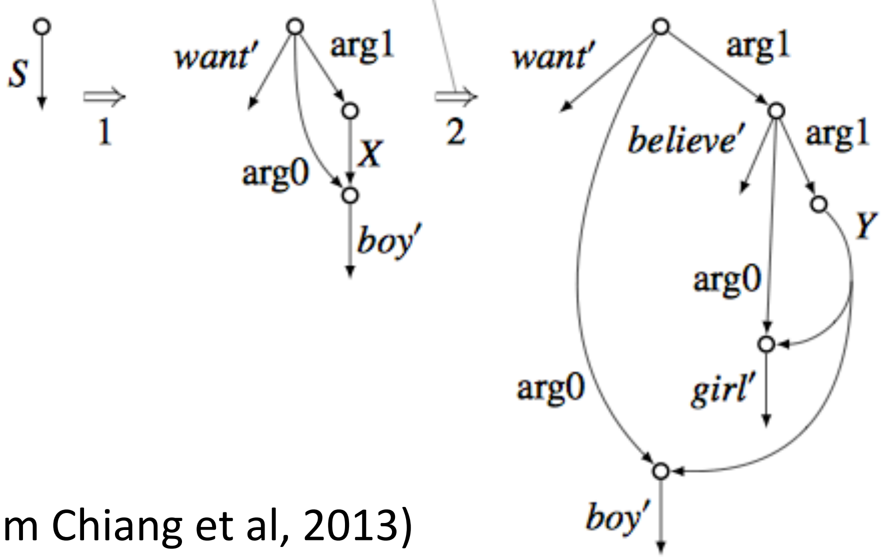
(figure from Chiang et al, 2013)

Hyperedge Replacement Grammar (HRG)

Grammar rules (some of them)



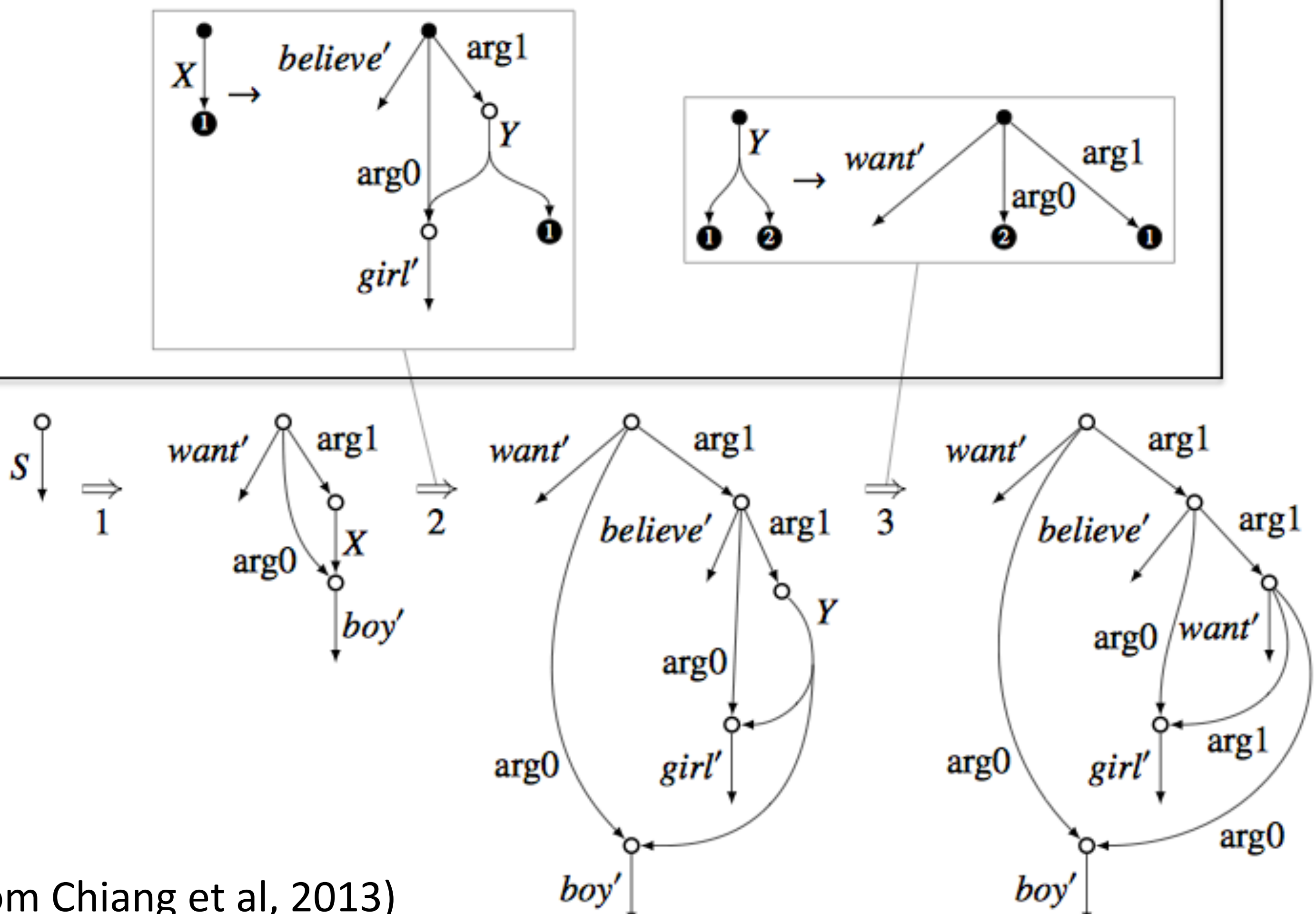




(figure from Chiang et al, 2013)

Hyperedge Replacement Grammar (HRG)

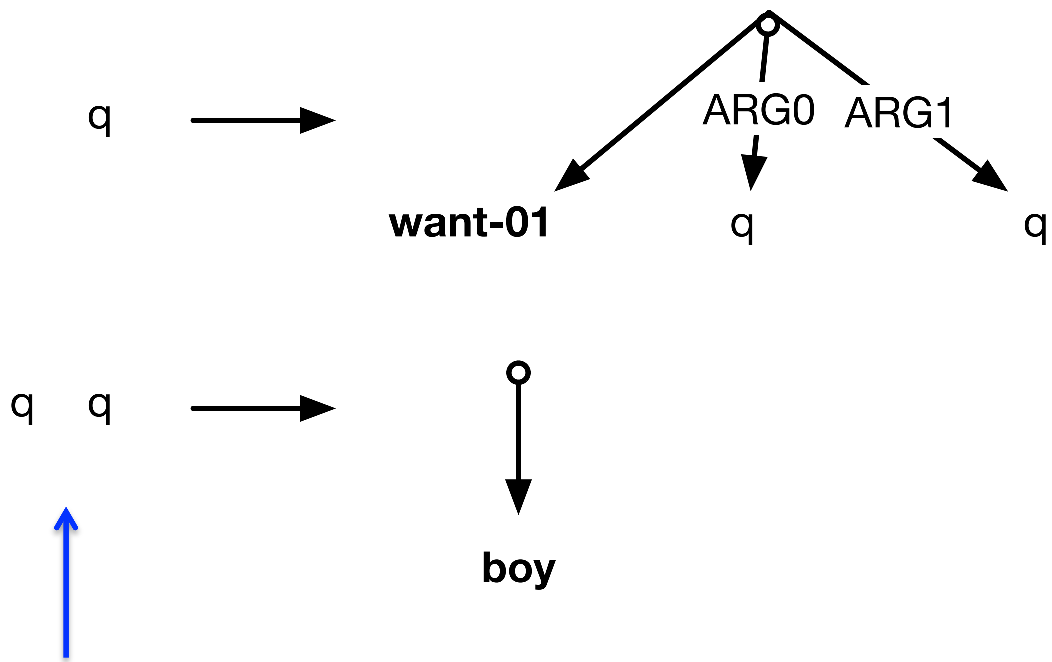
Grammar rules (some of them)



(figure from Chiang et al, 2013)

DAG Automata

(Kamimura and Slutzki, 1981. Quernheim and Knight, 2012)

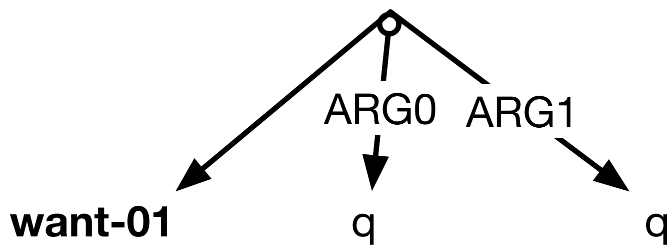


Rewrite rule
("explicit rule")

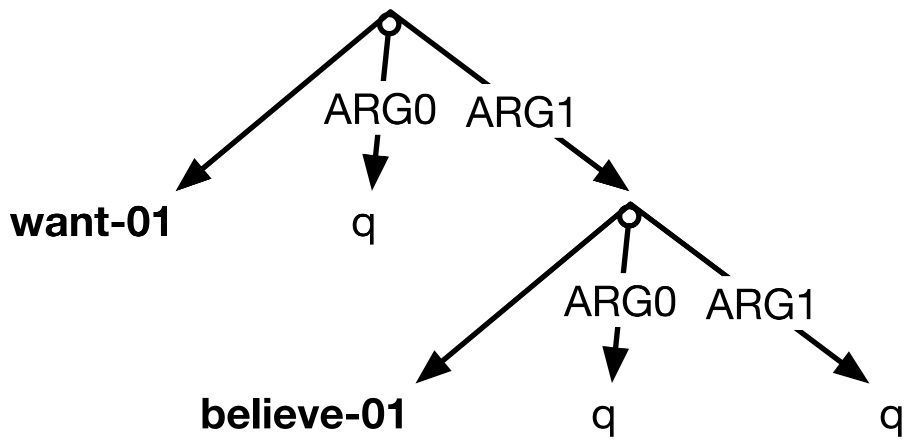
Merge rule
("implicit rule")

Two or more states can merge rule

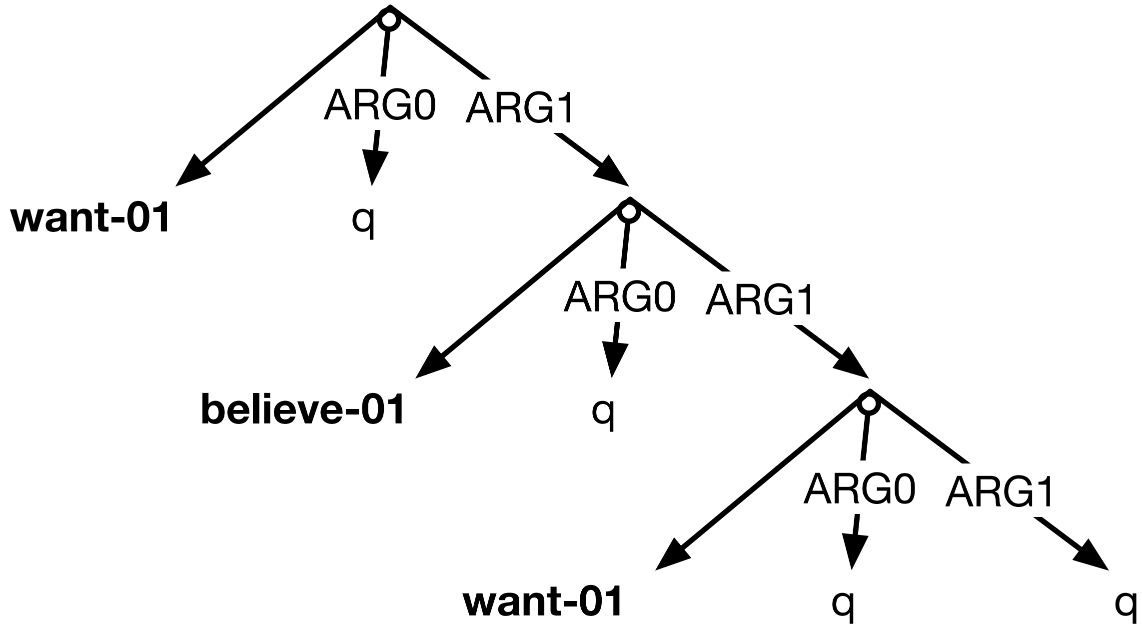
DAG Automata



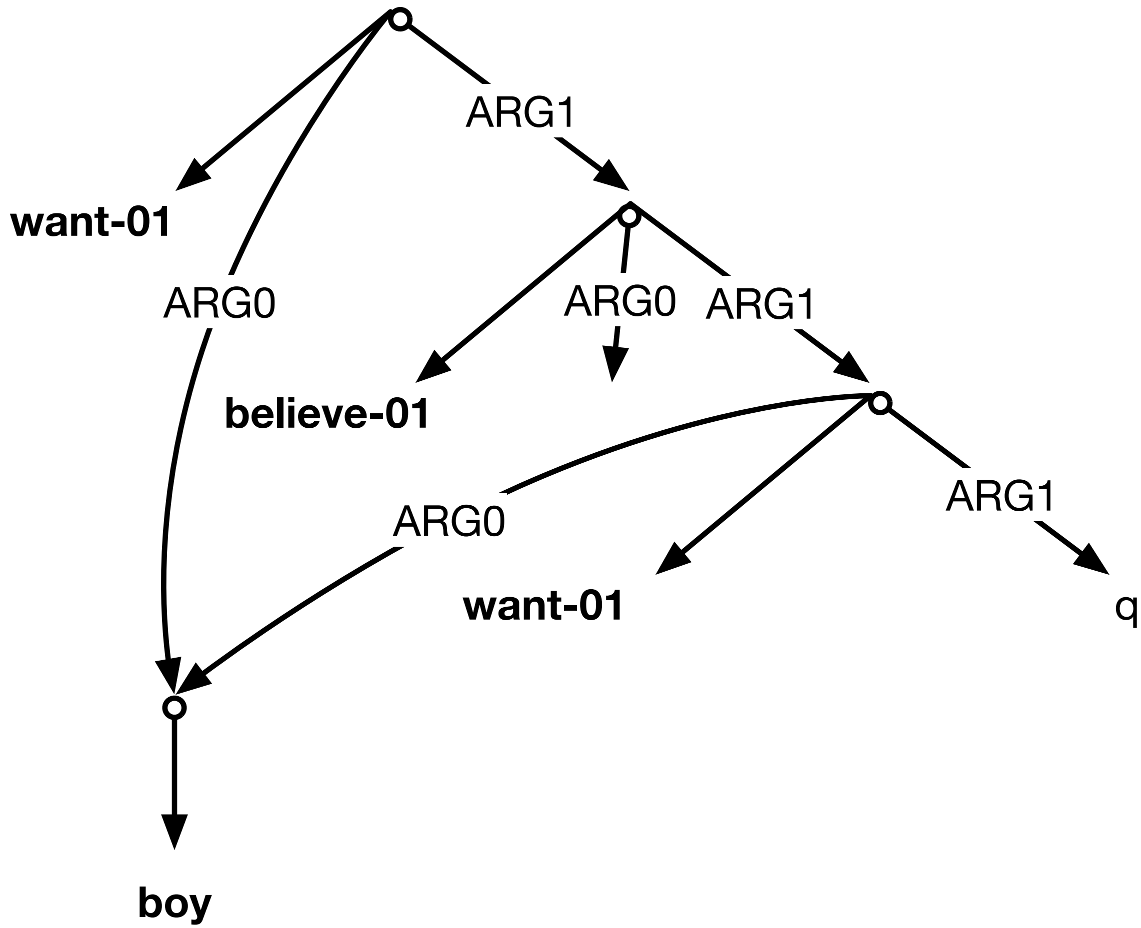
DAG Automata



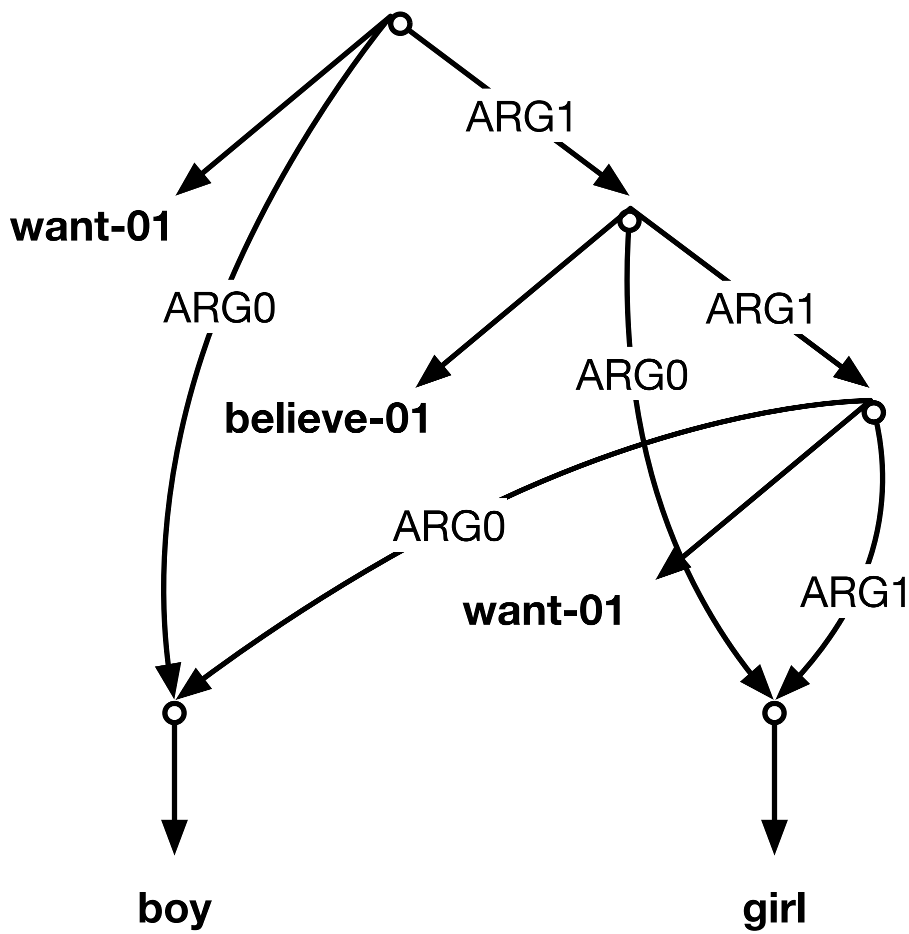
DAG Automata



DAG Automata



DAG Automata



Extensions

- Weighted and probabilistic grammars
- Synchronous grammars and transducers
  - Useful for building parsers, generators, and MT systems

Recent/Ongoing work

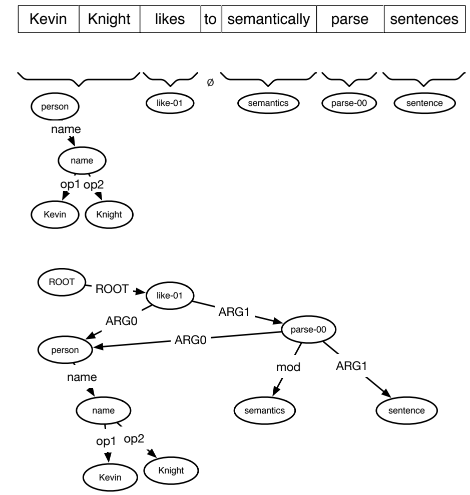
- Improved parsing algorithms (Chiang et al, 2013)
- Applications to parsing and generation (Braune et al, 2014) and MT (Jones et al, 2012)
- Implementations
  - Hyperedge replacement grammars: **Bolinas** (Chiang et al, 2013; Jones et al, 2012)
  - DAG automata: **DAGGER** (Quernheim & Knight, 2012)

Alignment

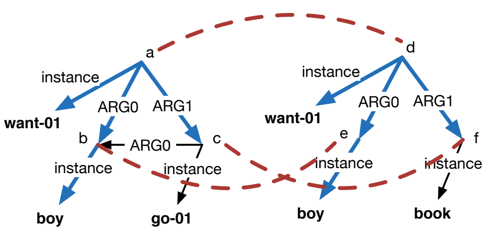
IAEA accepted North Korea 's proposal in November.

```
(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))
```

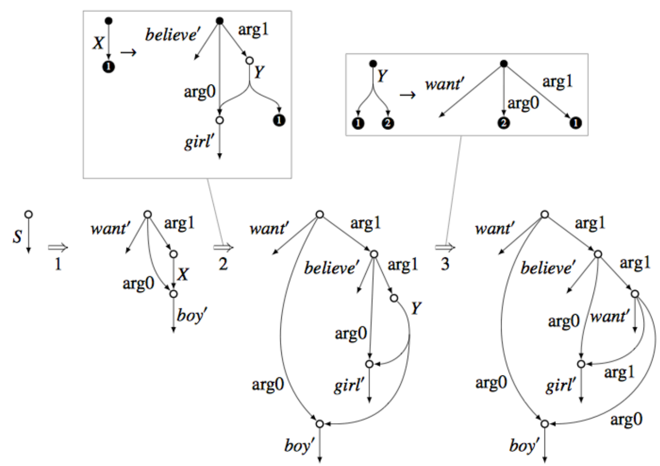
Parsing



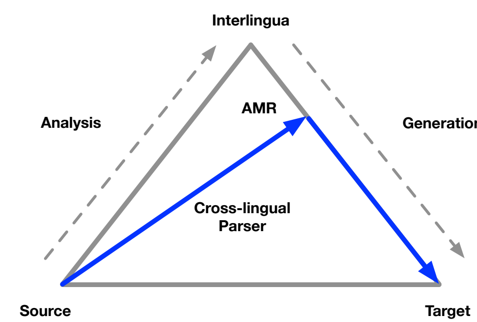
Evaluation



Graph grammars



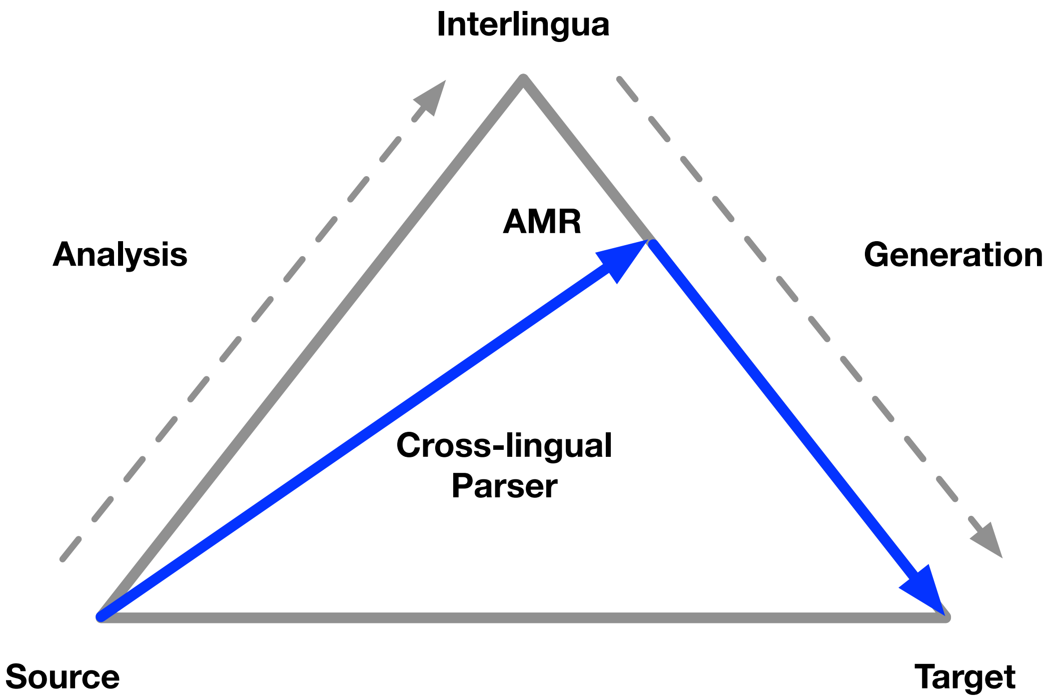
Applications



Applications

- Alignment
- Parsing
- Evaluation
- Graph Grammars and Automata
- **Applications**
  - MT, Summarization, Entity linking

Machine Translation



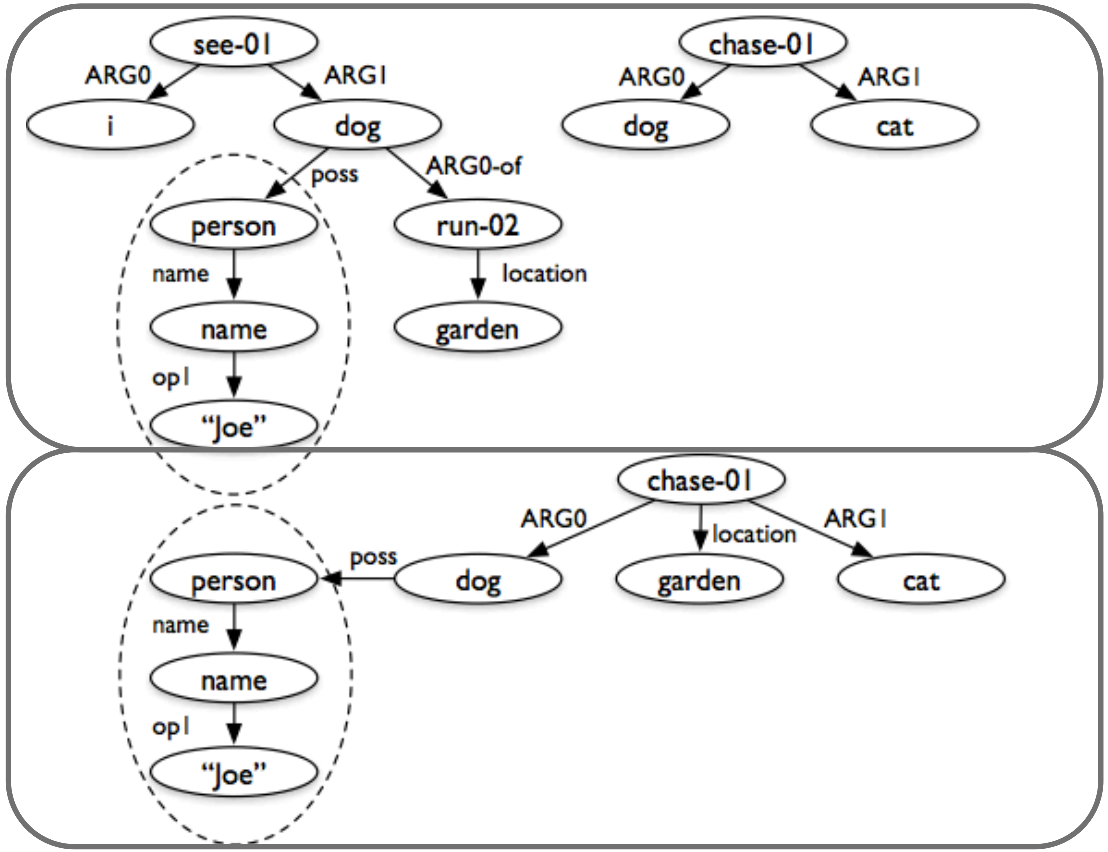
Summarization (Liu et al, NAACL 2015)

Document Sentences (input)

Sentence A: I saw Joe's dog, which was running in the garden.
Sentence B: The dog was chasing a cat.

Document AMRs (run parser)

Summary AMR (select nodes and edges)

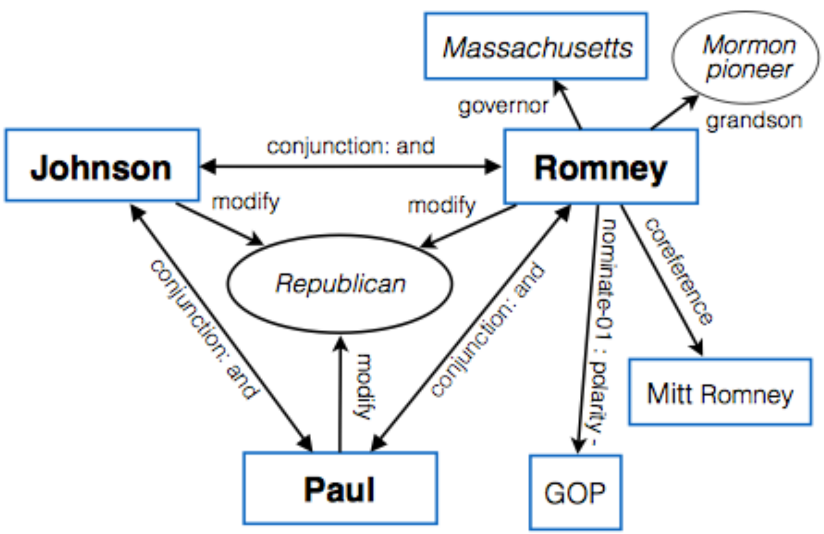


Summary (generate)

Summary: Joe's dog was chasing a cat in the garden.

Unsupervised Entity Linking with AMR (Pan et al, NAACL 2015)

- Link entity mentions in text to knowledge base
- Look at context to disambiguate mention
- Uses AMR graphs as context to build knowledge networks:



AMR context performs much better than SRL context for unsupervised entity linking

AMR at NAACL 2015

- Talks
  - Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, Noah A. Smith. "*Toward Abstractive Summarization Using Semantic Representations*"
  - Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, Kevin Knight. "*Unsupervised Entity Linking with Abstract Meaning Representation*
- Posters
  - Chuan Wang, Nianwen Xue, Sameer Pradhan. "*A Transition-based Algorithm for AMR Parsing*"
- Demonstrations
  - Lucy Vanderwende, Arul Menezes and Chris Quirk. "*An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus*"
  - Naomi Saphra and Adam Lopez. "*AMRICA: an AMR Inspector for Cross-language Alignments*"

Resources

- **AMR website**: http://amr.isi.edu
- **JAMR**: https://github.com/jflanigan/jamr
- **Transition-based parser**:

https://github.com/Juicechuan/AMRParsing/

- **Bolinas toolkit**:

http://www.isi.edu/publications/licensed-sw/bolinas/

- **DAGGER tookit**:

http://www.ims.uni-stuttgart.de/~daniel/dagger

```
(t / thank-01
      :ARG1 (y / you))
```

Thanks to: Miguel Ballesteros, David Chaing, Shay Cohen, Chris Dyer, Kevin Knight, Lingpeng Kong, Fei Liu, Noah Smith, Sam Thomson, and Chuan Wang

The Zen of AMR

(with apologies to “The Zen of Python”)

```
(l / love-01
   :ARG0 (p / person
            :name (n / name :op1 "Theodore"))
   :ARG1 (s / system
            :ARG0-of (o / operate-01
                        :ARG1 (c / computer
                                 :poss p))))
```

One graph to rule them all.

```
(l / love-01
    :ARG0 (p / person
             :name (n / name :op1 "Theodore"))
    :ARG1 (s / system
             :ARG0-of (o / operate-01
                         :ARG1 (c / computer
                                  :poss p))))
```

One graph to rule them all.

Defragmented semantics in (basically) a rooted DAG.

```
(l / love-01
   :ARG0 (p / person
            :name (n / name :op1 "Theodore"))
   :ARG1 (s / system
            :ARG0-of (o / operate-01
                        :ARG1 (c / computer
                                 :poss p))))
```

One graph to rule them all.

Defragmented semantics in (basically) a rooted DAG.

AMR is not in the derivation business.

```
(l / love-01
   :ARG0 (p / person
            :name (n / name :op1 "Theodore"))
   :ARG1 (s / system
            :ARG0-of (o / operate-01
                        :ARG1 (c / computer
                                 :poss p))))
```

Entity and event variables give coreference for free.

```
(l / love-01
   :ARG0 (p / person
            :name (n / name :op1 "Theodore"))
   :ARG1 (s / system
            :ARG0-of (o / operate-01
                        :ARG1 (c / computer
                                 :part-of p))))
```

Deep is better than shallow.
(Paraphrases should have the same AMR.)

…but practicality (for annotation) beats purity.

Theodore is in love with his computer's operating system.
Theodore loves his computer's operating system.
The operating system of Theodore's computer is loved by him.

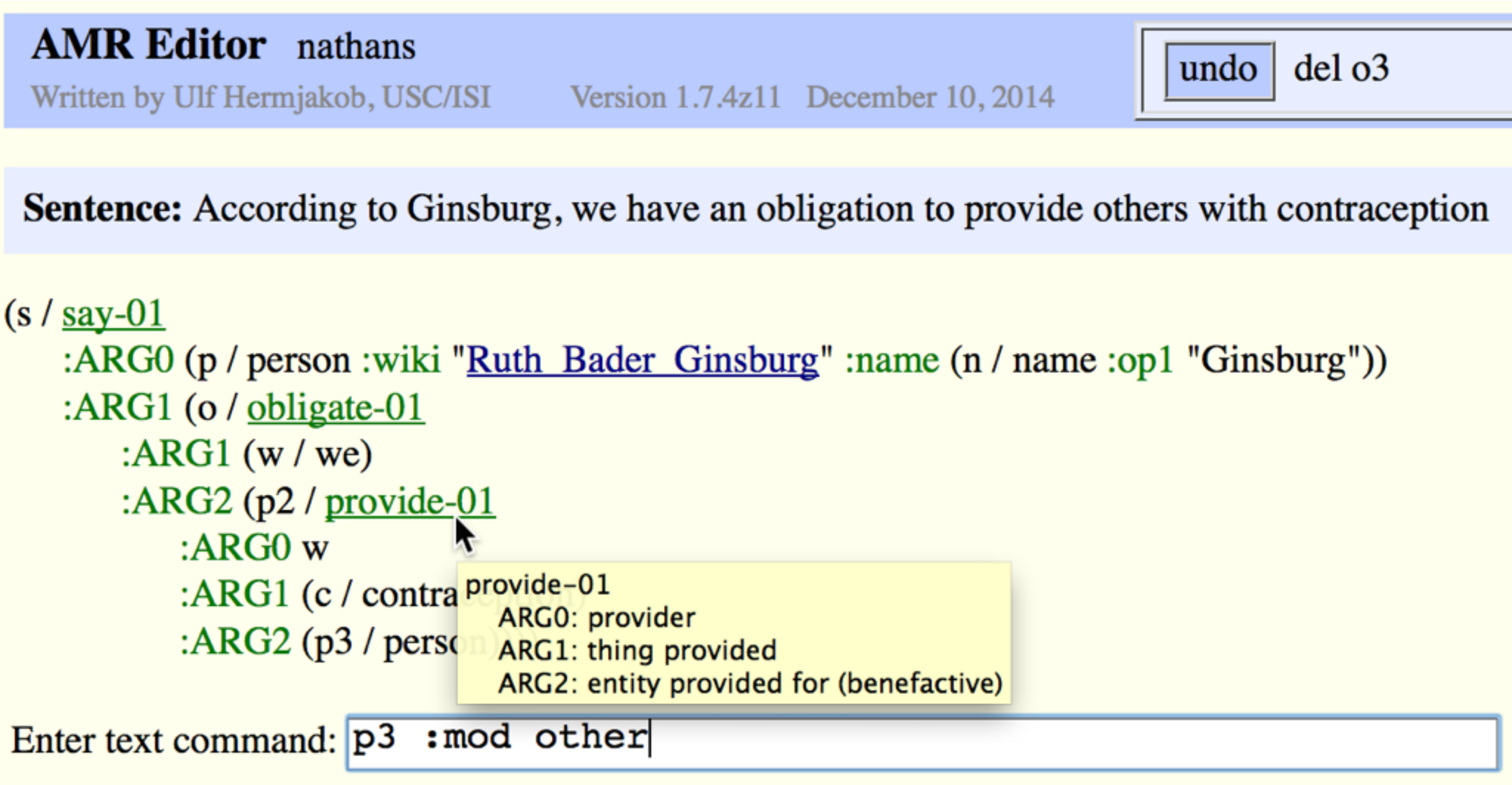
~~Theodore adores the operating system in his computer.~~
~~Theodore is romantically involved with his operating system.~~

It doesn’t have to be an interlingua,
so long as it is more logical than English.

There should be one and preferably only one obvious way to AMR it.

AMR is a canonical form.

The annotator experience matters.


AMR Editor nathans
Written by Ulf Hermjakob, USC/ISI Version 1.7.4z11 December 10, 2014
undo
del o3
Sentence: According to Ginsburg, we have an obligation to provide others with contraception
(s / say-01
:ARG0 (p / person :wiki "Ruth_Bader_Ginsburg" :name (n / name :op1 "Ginsburg"))
:ARG1 (o / obligate-01
:ARG1 (w / we)
:ARG2 (p2 / provide-01
:ARG0 w
:ARG1 (c / contra
:ARG2 (p3 / perso
provide-01
ARG0: provider
ARG1: thing provided
ARG2: entity provided for (benefactive)
Enter text command: p3 :mod other

how

- :ARGx
  - **How much** does this cost? `(cost-01 :ARG2 amr-unknown)` Example
  - **How** did you solve the problem? `(solve-01 :ARG2 amr-unknown)` Example
- :degree
  - **How** big was the dog that bit you? `(big :degree amr-unknown :domain :dog)` Example
  - **How beautiful** you are. [= You are so beautiful.] `(beautiful **:degree so** :domain you)` Example
- :manner
  - **How** did you get him to help you? `(get-04 :manner amr-unknown)` Example
  - The only thing that surprises me is **how rapidly** this is happening. `(surprise-01 :ARG0 (rapid :manner-of this))` Example
- :quant
  - **How tall** are you? `(tall **:quant amr-unknown** :domain you)` Example
  - **How old** is your father? `(father **:age amr-unknown** :poss you)` Example

however

- contrast-01 (most cases)
  - John, however, stayed at home. `(contrast-01 :ARG2 stay-01)` Example
- :concession
  - It started to rain; however, the game continued. `(start-01 :ARG1 rain-01 :concession-of continue-01)` Example

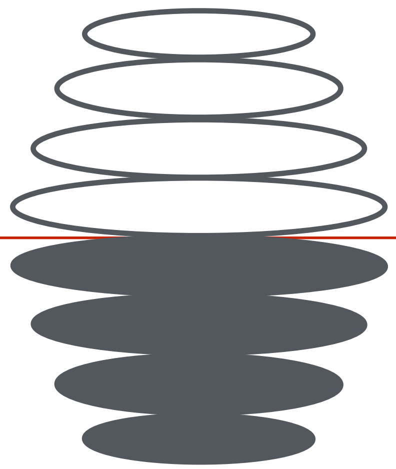
imperative

- :mode imperative
  - Go home! `(go-02 :mode imperative :ARG0 you :destination home)` Example
  - Let's go home. `(go-02 :mode imperative :ARG0 we :destination home)` Example
  - Come on folks!! `(come-25 :mode imperative :ARG1 folk)` Example
- :mode imperative :polite +
  - **Please** close the door. `(close-01 :mode imperative :polite +)` Example
  - **Could** you close the door? `(close-01 :mode imperative :polite +)` Example

Morphosyntactic sugar
is considered unhealthy.

```
(d / decide-01
    :ARG0 (h / he)
    :mod (p / politics)
    :manner (t / think-01 :polarity -
                :ARG0 h))
```

He made a thoughtless political decision.



Verbs (events)
are better than nouns (plain concepts).

Adjectives are better than adverbs.

Adjective frames are better than `:mod`.

Core roles are better than non-core roles.

Anything is better than `:prep-X`!

```
(p / president)
```

the president

```
(p / person :ARG0-of (p2 / preside-01))
```

the person who presides

Verbs (events)
are better than nouns (plain concepts).

(Except when the meaning is noncompositional.)

Adjectives are better than adverbs.

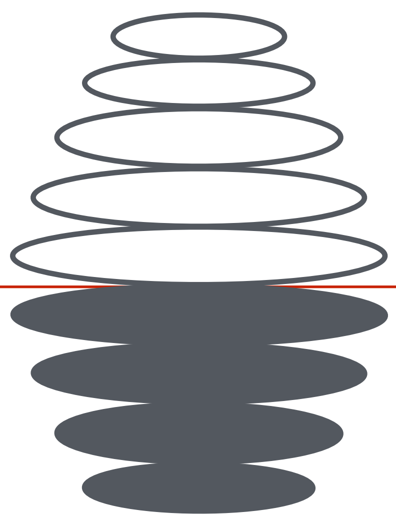
Adjective frames are better than `:mod`.

Core roles are better than non-core roles.

Anything is better than `:prep-X`!

```
(p / person
   :ARG0-of (h / have-org-role-91
                 :ARG1 (…Russia…)
                 :ARG2 (p2 / president)))
```

the Russian president



Occasionally you need to hallucinate to fill in the gaps.

```
(m / meal
   :ARG1-of (c / cost-01
                 :ARG2 (m2 / monetary-quantity
                           :quant 3
                           :unit (d / dollar))))
```

a $3 meal

a meal that costs $3

```
(g / government-organization
   :ARG0-of (g2 / govern-01
                :ARG1 (…China…)))
```

the Chinese government
the government of China

A government is a **government-organization** that governs.

```
(g / government-organization
   :ARG0-of (g2 / govern-01)
   :mod (f / federal))
```

the federal government

The sentence is a starting point.
It should not be a straightjacket.

In the future, AMR will pay better attention to cross-sentence/discourse phenomena.

AMR does not compromise for algorithmic expedience. (Algorithms will have to catch up!)

http://amr.isi.edu

AMR Bibliography

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer & Nathan Schneider. 2013. **Abstract Meaning Representation for sembanking**. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. Sofia, Bulgaria: Association for Computational Linguistics. http://www.aclweb.org/anthology/W13-2322.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer & Nathan Schneider. 2014. **Abstract Meaning Representation (AMR) 1.2 Specification**. https://github.com/amrisi/amr-guidelines/blob/b0fd2d6321ed4c9e9fa202b307cceeae36b8c25b/amr.md.

Bonial, Claire, Julia Bonn, Kathryn Conger, Jena D. Hwang & Martha Palmer. 2014. **PropBank: semantics of new predicate types**. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 3013–3019. Reykjavík, Iceland: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2014/pdf/1012_Paper.pdf.

Braune, Fabienne, Daniel Bauer & Kevin Knight. 2014. **Mapping between English strings and reentrant semantic graphs**. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. Reykjavík, Iceland: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2014/pdf/1080_Paper.pdf.

Cai, Shu & Kevin Knight. 2013. **Smatch: an evaluation metric for semantic feature structures**. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 748–752. Sofia, Bulgaria: Association for Computational Linguistics. http://www.aclweb.org/anthology/P13-2131.

Chiang, David, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones & Kevin Knight. 2013. **Parsing graphs with hyperedge replacement grammars**. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 924–932. Sofia, Bulgaria: Association for Computational Linguistics. http://www.aclweb.org/anthology/P13-1091.

Flanigan, Jeffrey, Sam Thomson, Jaime Carbonell, Chris Dyer & Noah A. Smith. 2014. **A discriminative graph-based parser for the Abstract Meaning Representation**. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 1426–1436. Baltimore, Maryland, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/P14-1134.

Jones, Bevan, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann & Kevin Knight. 2012. **Semantics-based machine translation with hyperedge replacement grammars**. *Proceedings of COLING 2012*, 1359–1376. Mumbai, India: The COLING 2012 Organizing Committee. http://www.aclweb.org/anthology/C12-1083.

Knight, Kevin, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer & Nathan Schneider. 2014. **Abstract Meaning Representation (AMR) Annotation Release 1.0**. Philadelphia, Pennsylvania, USA: Linguistic Data Consortium. https://catalog.ldc.upenn.edu/LDC2014T12.

Koller, Alexander. 2015. **Semantic construction with graph grammars**. *Proceedings of the 11th International Conference on Computational Semantics*, 228–238. London, UK: Association for Computational Linguistics. http://www.aclweb.org/anthology/W15-0127.

Langkilde, Irene & Kevin Knight. 1998. **Generation that exploits corpus-based statistical knowledge**. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, 704–710. Montreal, Quebec, Canada: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=980963.

Liu, Fei, Jeffrey Flanigan, Sam Thomson, Norman Sadeh & Noah A. Smith. 2015. **Toward abstractive summarization using semantic representations**. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1077–1086. Denver, Colorado: Association for Computational Linguistics. http://www.aclweb.org/anthology/N15-1114.

Palmer, Martha, Daniel Gildea & Paul Kingsbury. 2005. **The Proposition Bank: an annotated corpus of semantic roles**. *Computational Linguistics* 31(1). 71–106. doi:10.1162/0891201053630264. http://dx.doi.org/10.1162/0891201053630264.

Pan, Xiaoman, Taylor Cassidy, Ulf Hermjakob, Heng Ji & Kevin Knight. 2015. **Unsupervised entity linking with Abstract Meaning Representation**. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1130–1139. Denver, Colorado: Association for Computational Linguistics. http://www.aclweb.org/anthology/N15-1119.

Pourdamghani, Nima, Yang Gao, Ulf Hermjakob & Kevin Knight. 2014. **Aligning English strings with Abstract Meaning Representation graphs**. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 425–429. Doha, Qatar: Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1048.

Pust, Michael, Ulf Hermjakob, Kevin Knight, Daniel Marcu & Jonathan May. 2015. **Using syntax-based machine translation to parse English into Abstract Meaning Representation**. arXiv: 1504.06665 [cs]. http://arxiv.org/abs/1504.06665.

Quernheim, Daniel & Kevin Knight. 2012. **Towards probabilistic acceptors and transducers for feature structures**. *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 76–85. Jeju, Republic of Korea: Association for Computational Linguistics. http://www.aclweb.org/anthology/W12-4209.

Saphra, Naomi & Adam Lopez. 2015. **AMRICA: an AMR Inspector for Cross-language Alignments**. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 36–40. Denver, Colorado: Association for Computational Linguistics. http://www.aclweb.org/anthology/N15-3008.

Schneider, Nathan, Vivek Srikumar, Jena D. Hwang & Martha Palmer. 2015. **A hierarchy with, of, and for preposition supersenses**. *Proceedings of The 9th Linguistic Annotation Workshop*, 112–123. Denver, Colorado, USA: Association for Computational Linguistics. http://www.aclweb.org/anthology/W15-1612.

Vanderwende, Lucy, Arul Menezes & Chris Quirk. 2015. **An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus**. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 26–30. Denver, Colorado: Association for Computational Linguistics. http://www.aclweb.org/anthology/N15-3006.

Wang, Chuan, Nianwen Xue & Sameer Pradhan. 2015. **A Transition-based algorithm for AMR parsing**. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 366–375. Denver, Colorado: Association for Computational Linguistics. http://www.aclweb.org/anthology/N15-1040.

Werling, Keenon, Gabor Angeli & Christopher D. Manning. 2015. **Robust subgraph generation improves Abstract Meaning Representation parsing**. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. Beijing, China. To appear.

Xue, Nianwen, Ondrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova & Xiuhong Zhang. 2014. **Not an interlingua, but close: comparison of English AMRs to Chinese and Czech**. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 1765–1772. Reykjavík, Iceland: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2014/pdf/384_Paper.pdf.

AMR Editor Command Line

Editing

- **Adding stuff**
  - Start the AMR: `top concept`
  - Attach a new concept under a variable v: `v :REL concept`
  - Attach a new NE under a variable: `v :REL type Parts Of Name`
  - Add a new relation between two existing variables: `v :REL v2`
- **Changing stuff**
  - Move a variable under another one: `v :REL v2-`
  - Change the relation between two variables: `v :NEWREL v2-`
  - Select a PropBank frame: Add as a plain concept, click on the blue link, then click on the frame you want to add a sense number (the link in the AMR will turn green). (If you know the sense number in advance you can enter it as the concept name.)
  - Rename a concept: `rc v new-concept`
  - Delete the primary occurrence of a variable and everything under it that is not used elsewhere: `del v`
  - Delete dummy nodes (unused suggestions from preprocessor): `dd`
- Undo last action: `u`
- Save changes: `save`
- Save and continue to the next sentence: click the button

Documentation

- Open the list of non-core roles and special frames: `roles`
- Open the list of entity types: `types`
- Open the AMR Dictionary: `dict`
- Open the AMR Guidelines: `guidelines`
- Search all release AMRs: `rs query`