A constrained graph algebra for semantic parsing with AMRs

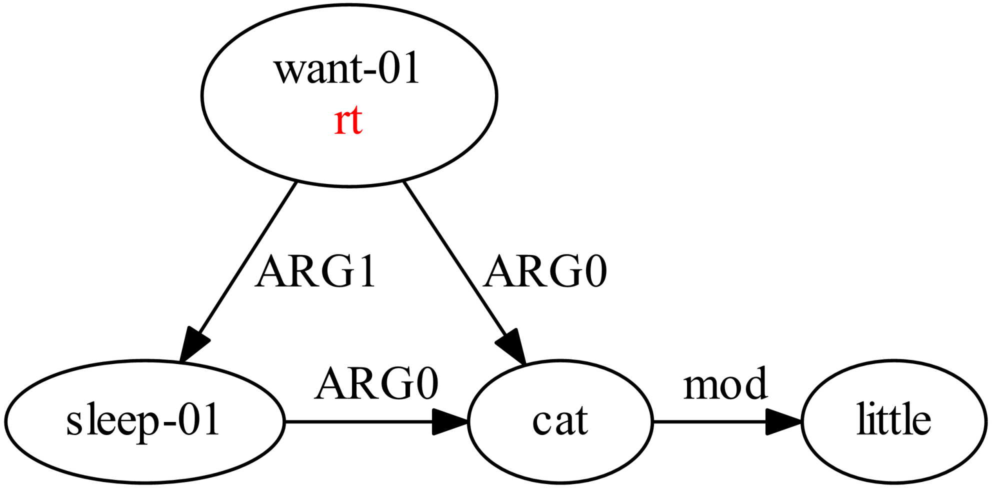
IWCS 2017

Jonas Groschwitz[#+], Meaghan Fowlie[#],
Alexander Koller[#], Mark Johnson[+]

#: Saarland University, +: Macquarie University

Abstract Meaning Representation (AMR)

- **Semantic representations** of sentences.
- **Rooted graphs**.
- Graph nodes represent **concepts** of the sentence.
- Edges **relate** these concepts.

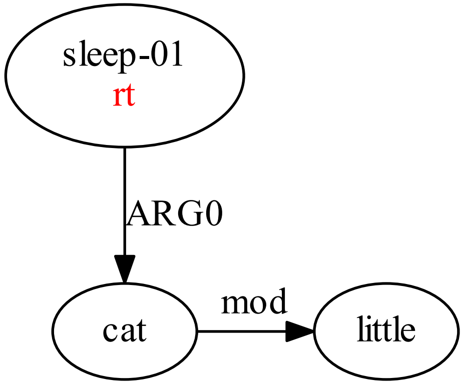


"The little cat wants to sleep."

Why AMRs

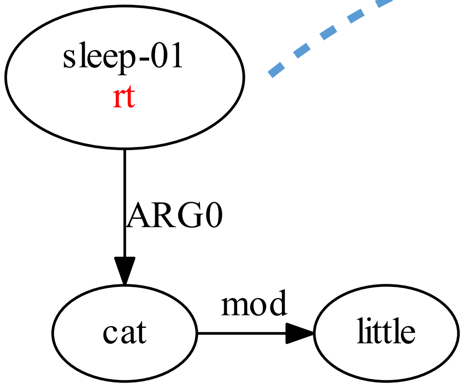
- AMRs are available, have big corpora, and make a good place to start looking at semantic parsing
- However, we will define operations for composing graphs that are general enough that we think the basic principle could be applied to other domains

Dependency Parsing into AMRs



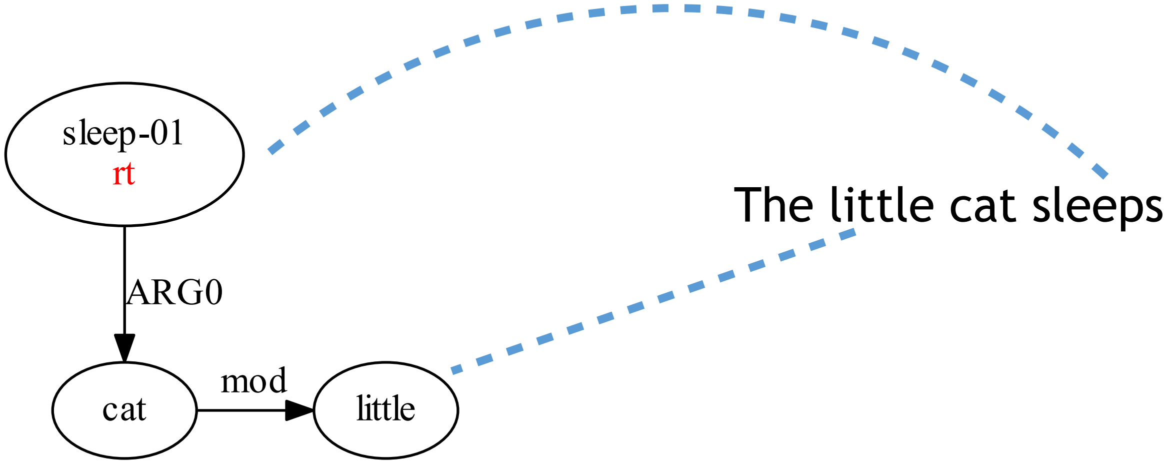
The little cat sleeps

Dependency Parsing into AMRs



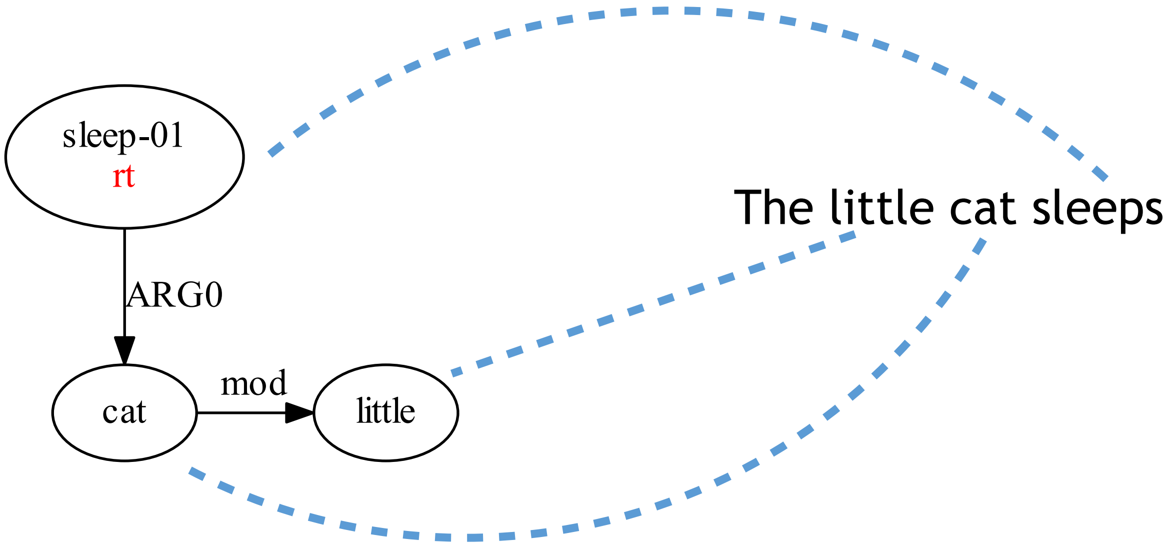
The little cat sleeps

Dependency Parsing into AMRs



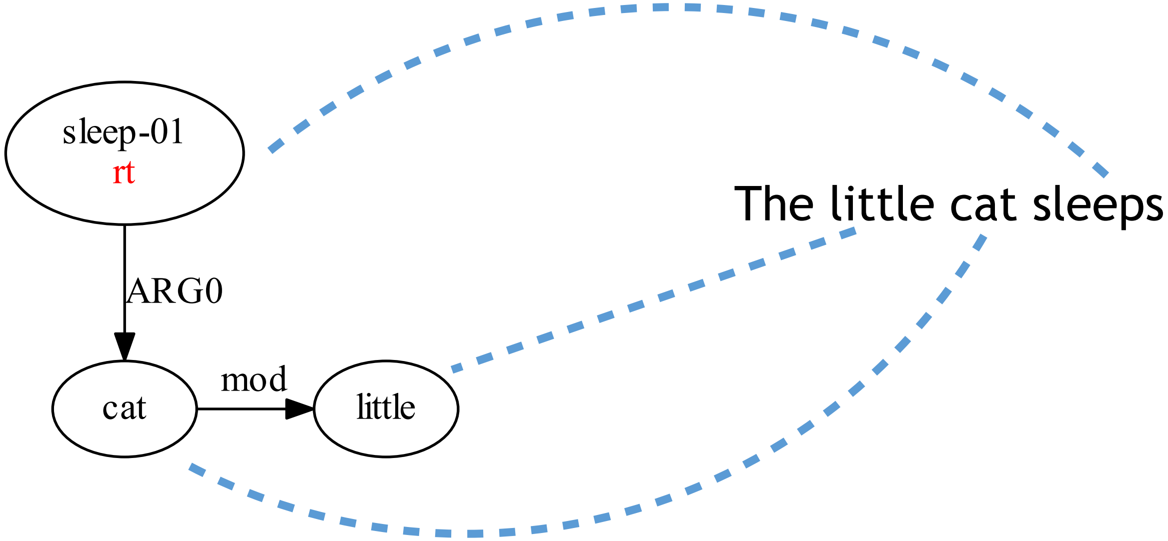
The little cat sleeps

Dependency Parsing into AMRs

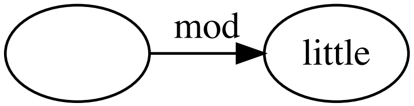


The little cat sleeps

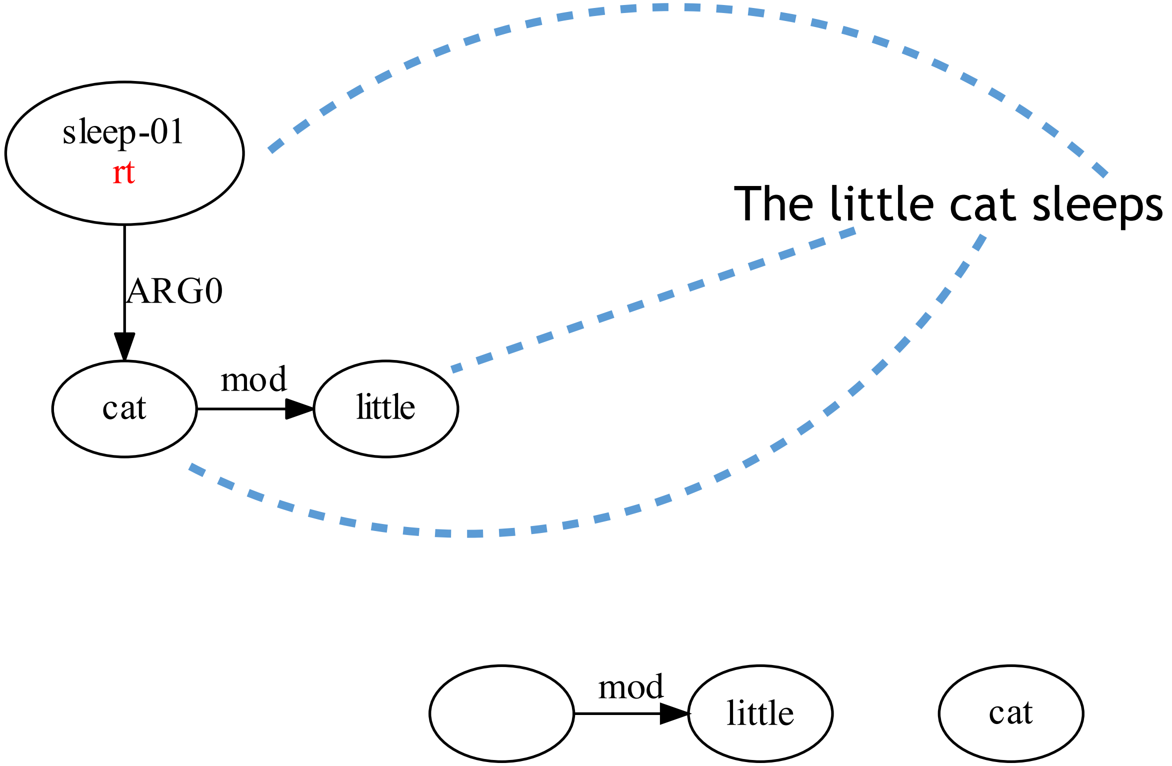
Dependency Parsing into AMRs



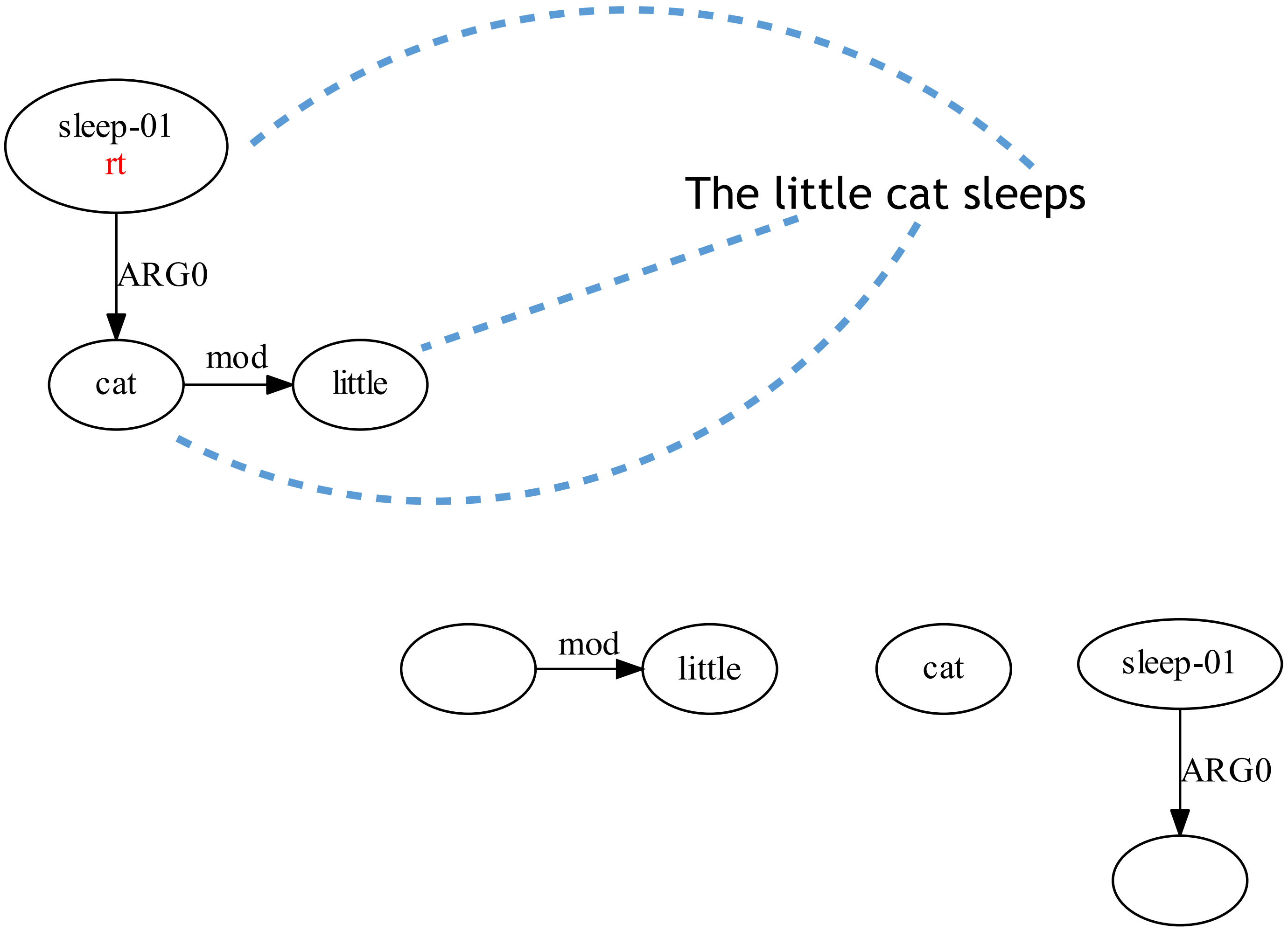
The little cat sleeps



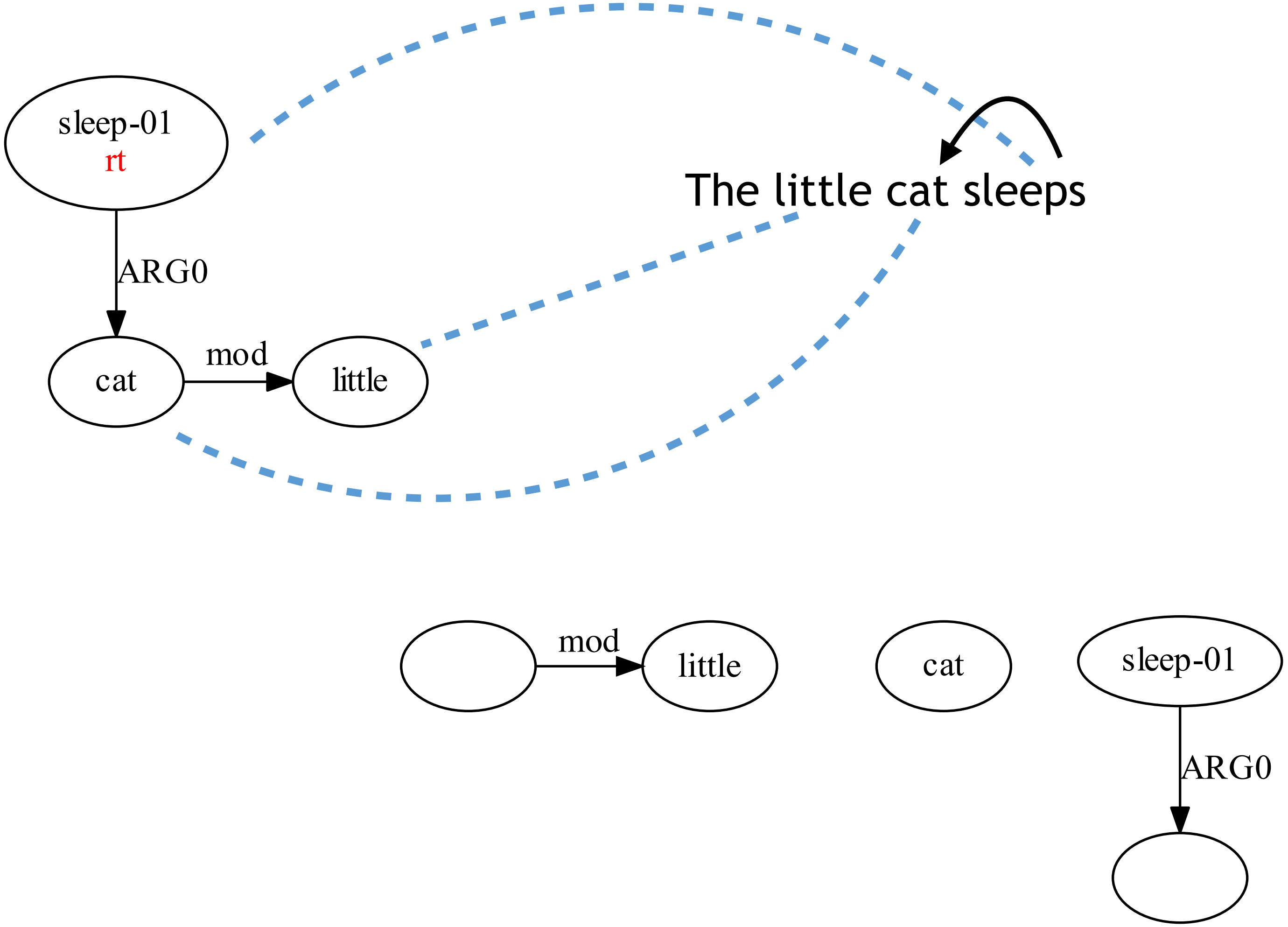
Dependency Parsing into AMRs



Dependency Parsing into AMRs

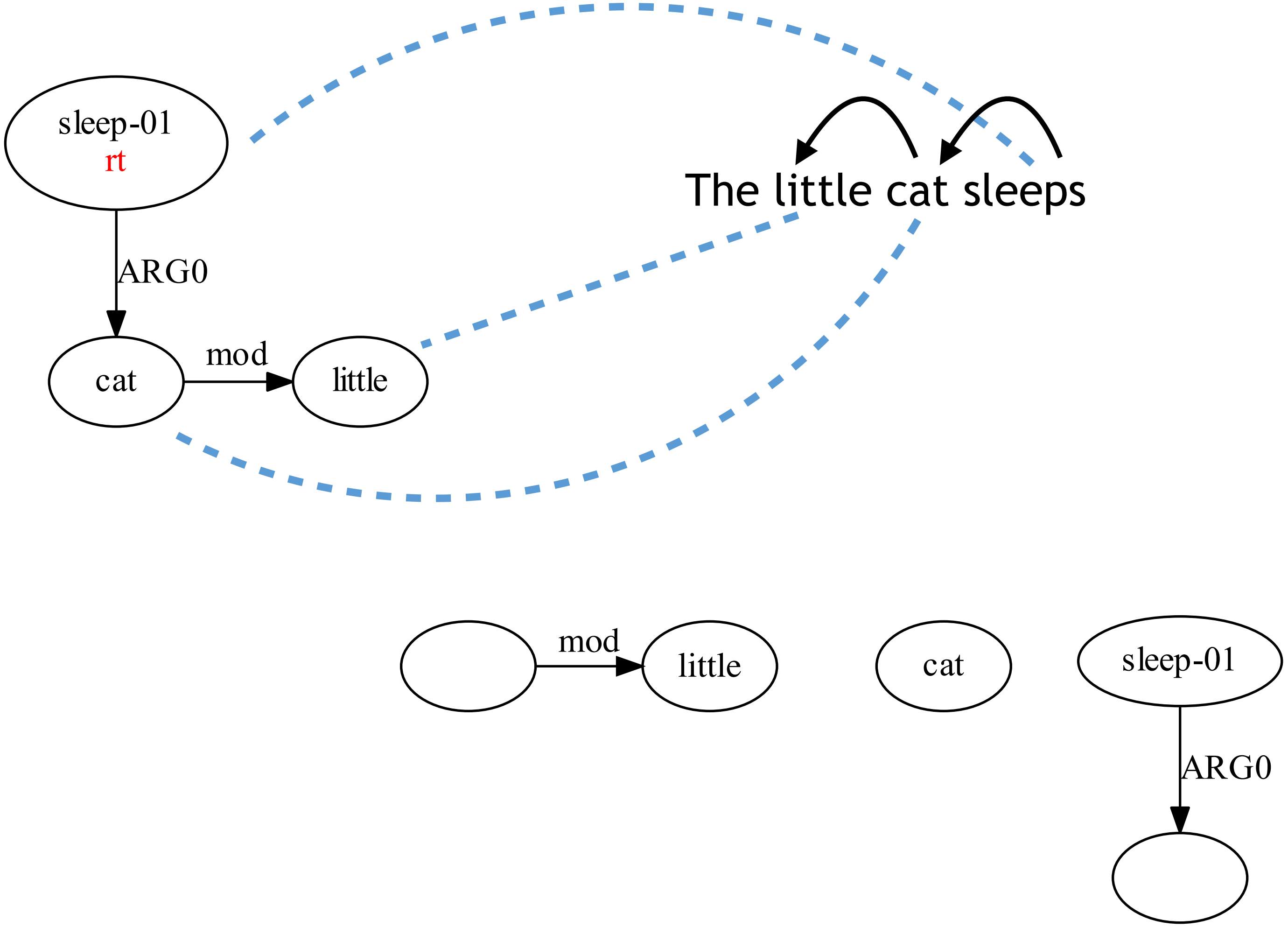


Dependency Parsing into AMRs



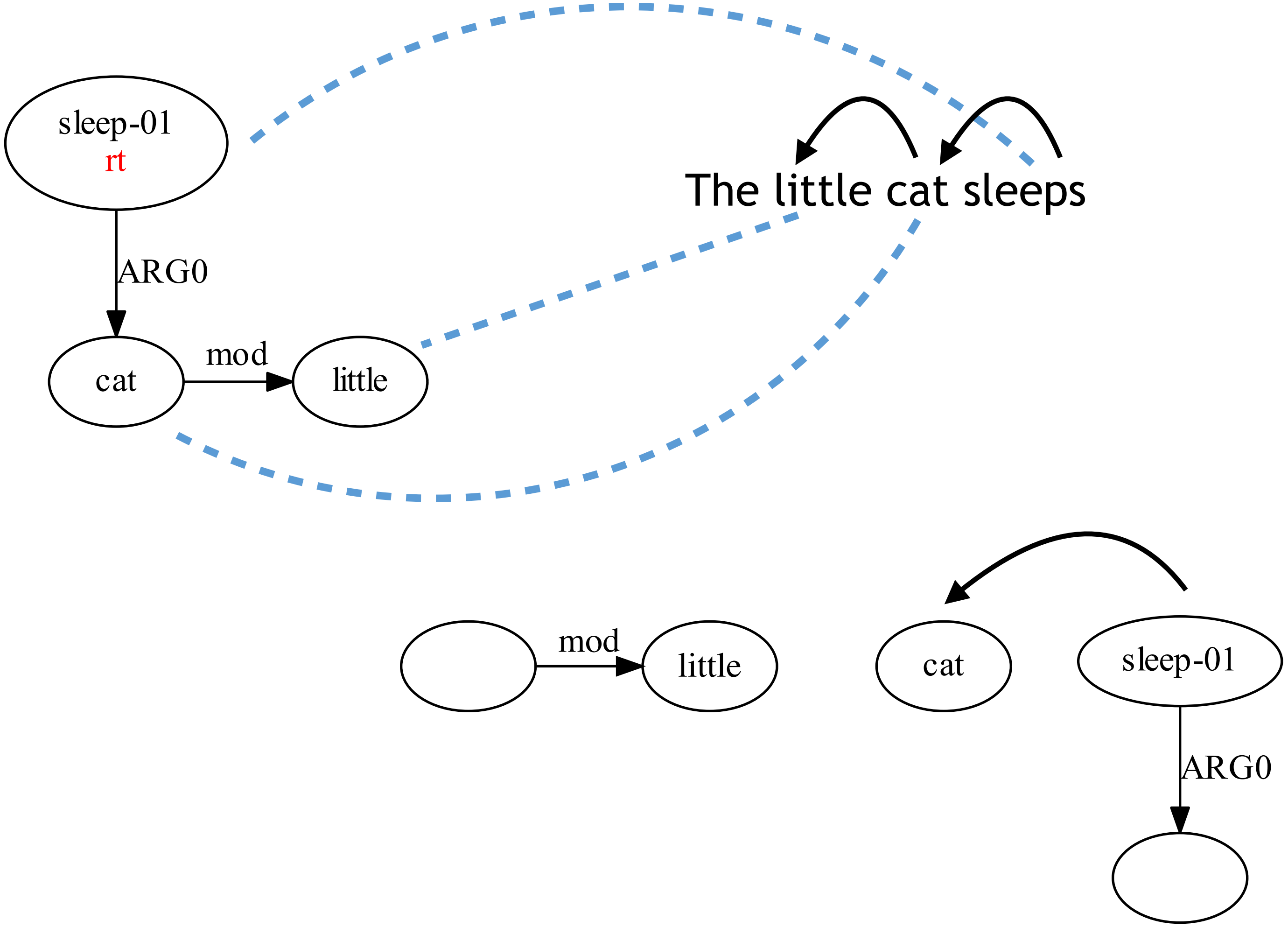
The little cat sleeps

Dependency Parsing into AMRs

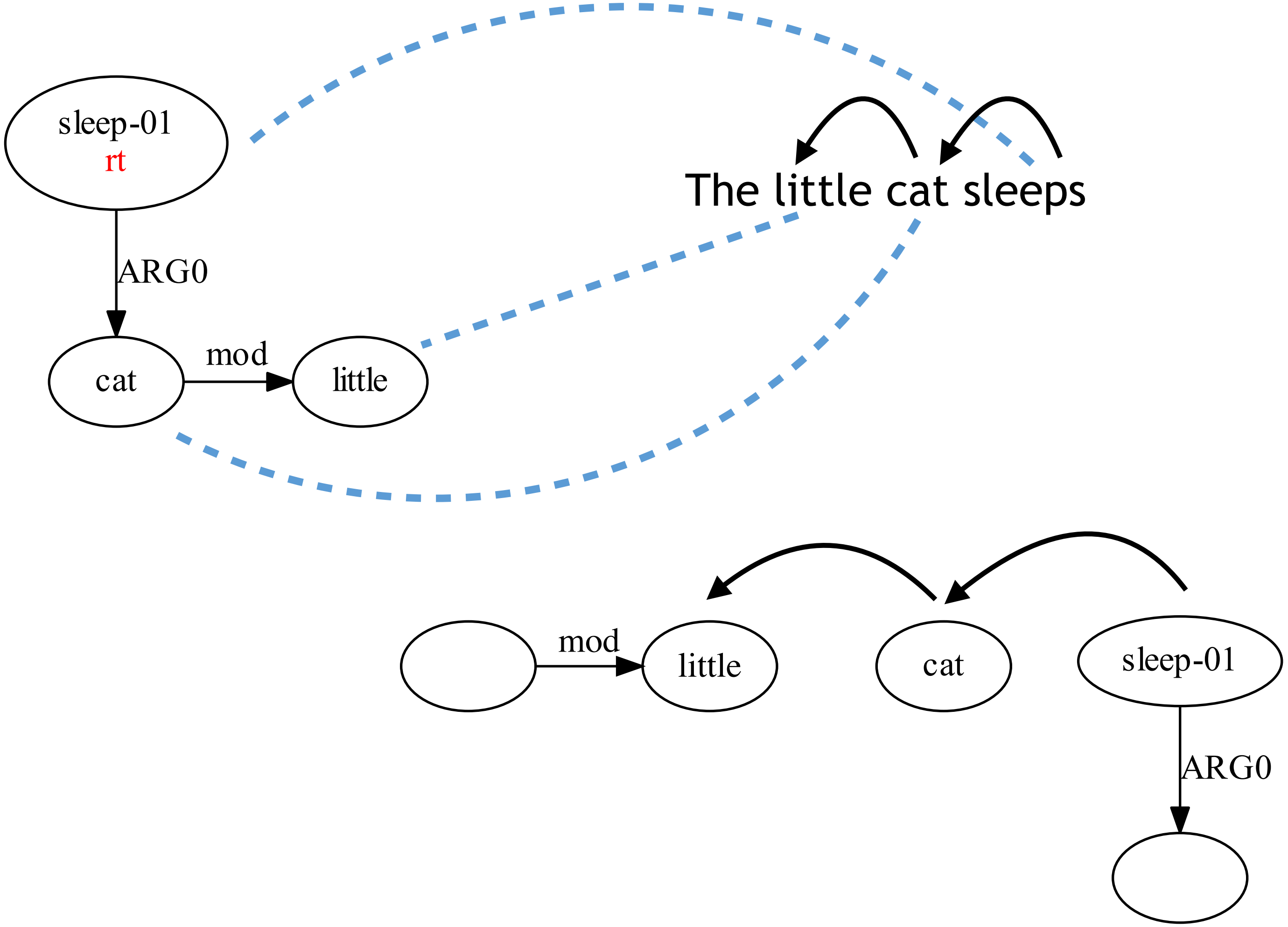


The little cat sleeps

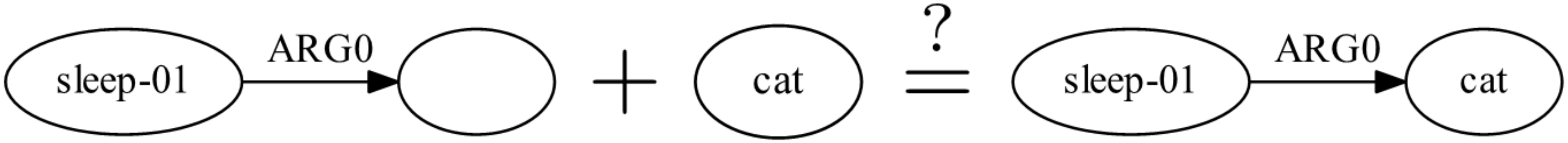
Dependency Parsing into AMRs



Dependency Parsing into AMRs

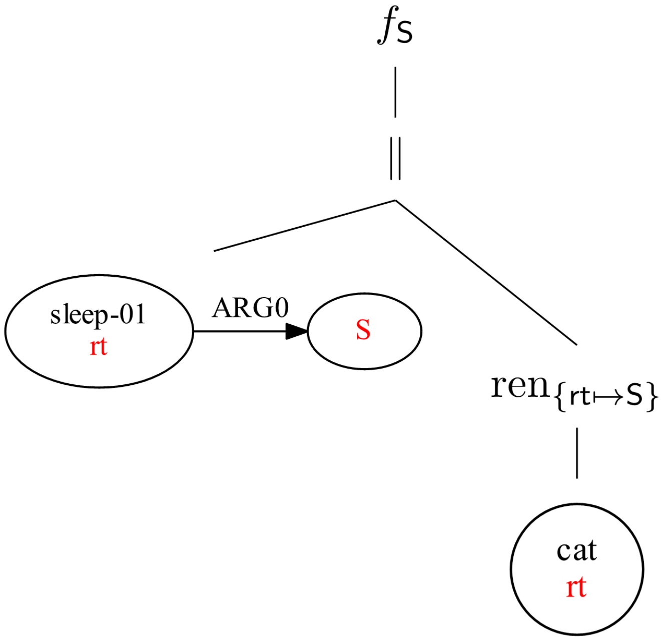


Building Graphs



Building Graphs

- Formally: **algebra**
- **terms** contain **symbols** representing **operations**

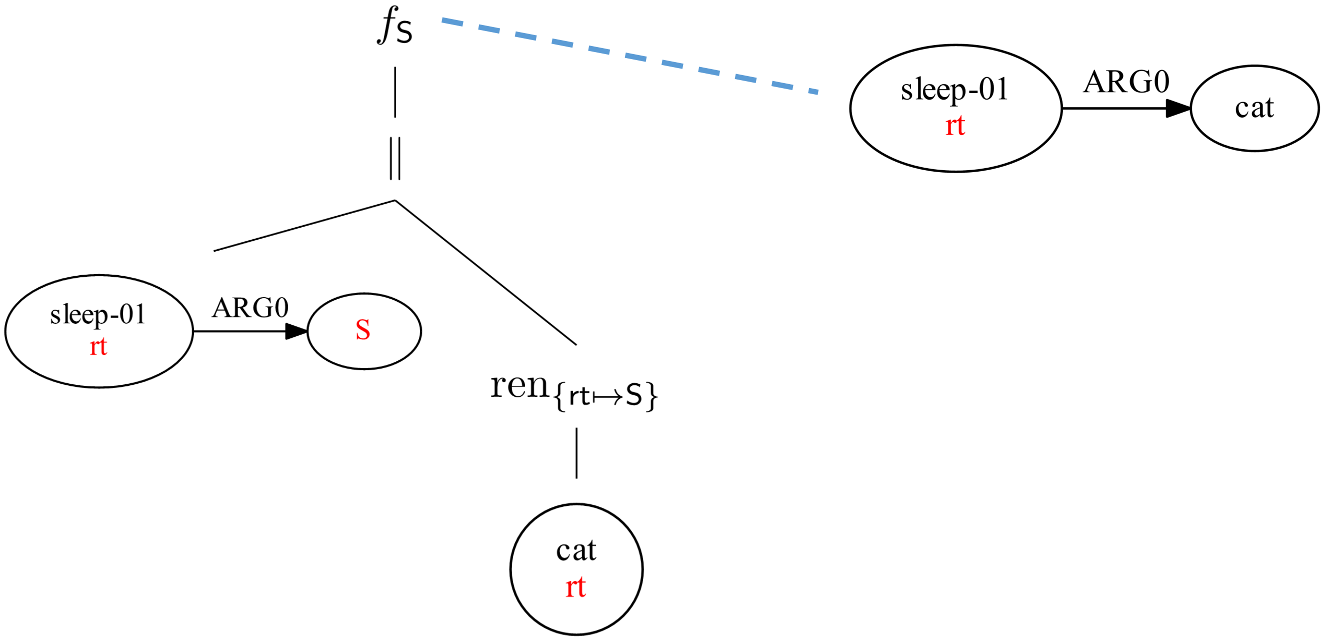


Training a system to find dependencies:

generate a lot of terms and try to find patterns.

Building Graphs

- Formally: **algebra**
- **terms** contain **symbols** representing **operations**



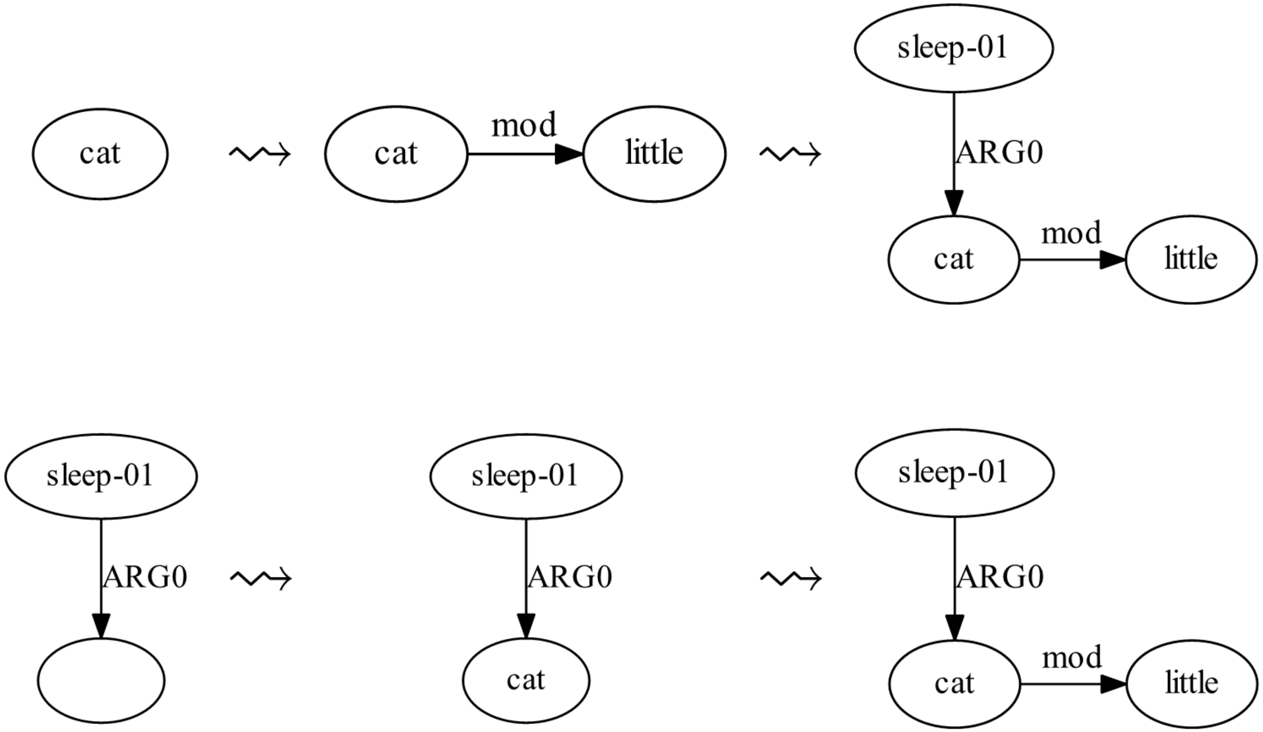
Training a system to find dependencies:

generate a lot of terms and try to find patterns.

Compositional complexity

There is more than one way to build a graph!

hidden compositional structure



Our Goal

An algebra that

1. has low compositional complexity,
2. produces consistent, meaningful terms

Our Goal

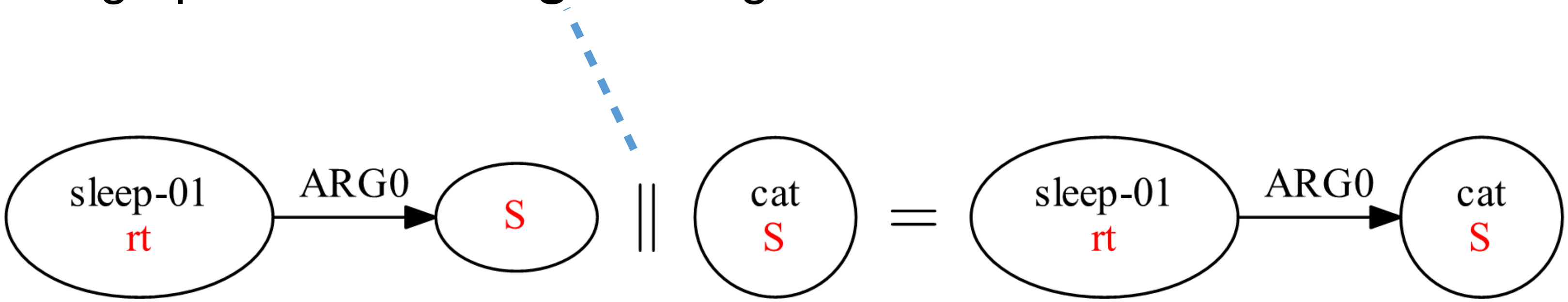
An algebra that

1. has low compositional complexity,
2. produces consistent, meaningful terms

➢ Use linguistics!

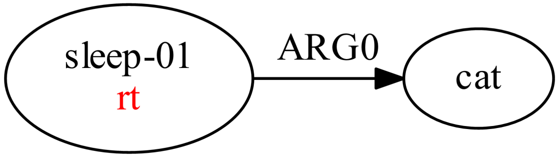
HR algebra (Courcelle, 1993)

- Mark some graph nodes with **source names**.
- ***S-graphs:*** graphs with source names
- S-graphs can be **merged** along common source names.

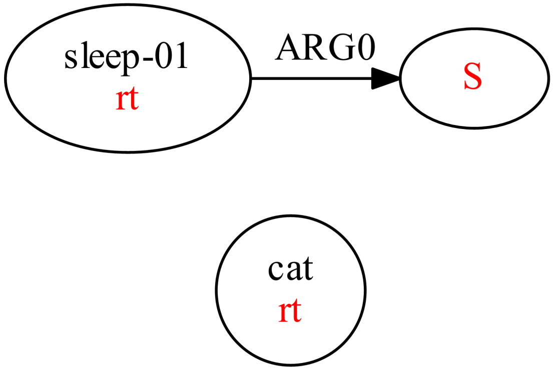


HR algebra (Courcelle, 1993)

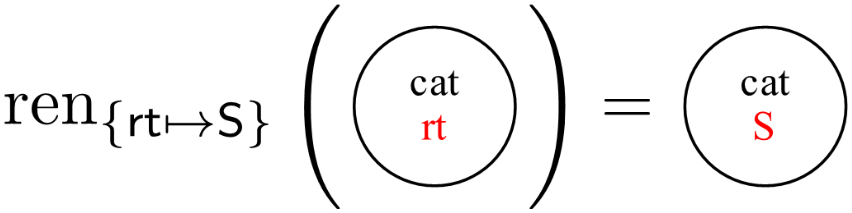
- Source names are introduced in lexical constants:



“The cat sleeps.”

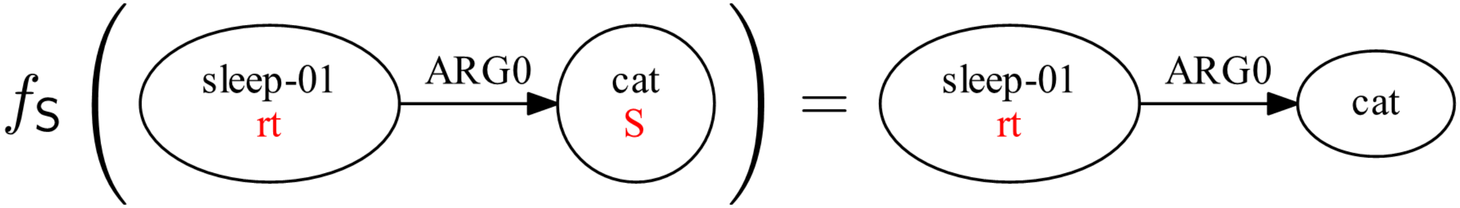


- Can **rename** source names:



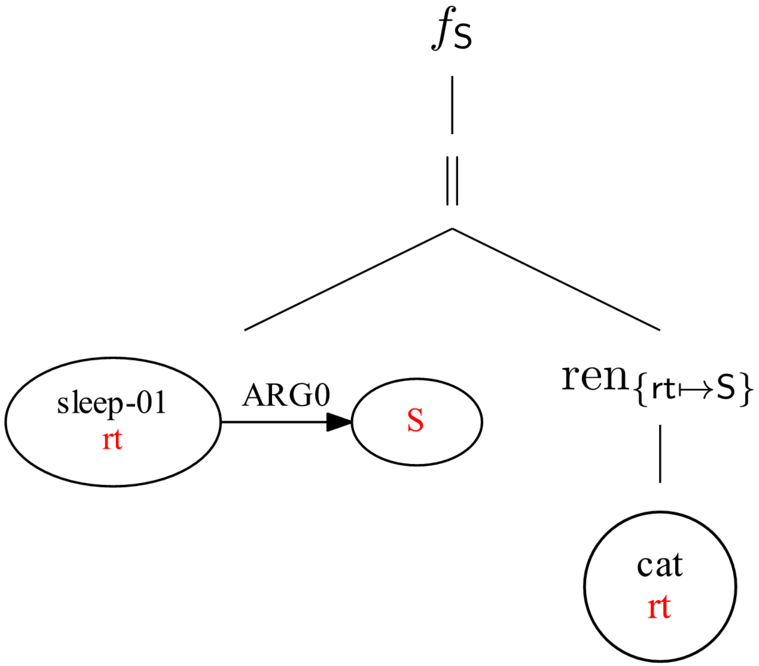
HR algebra (Courcelle, 1993)

- Can **forget** source names when we no longer need them:



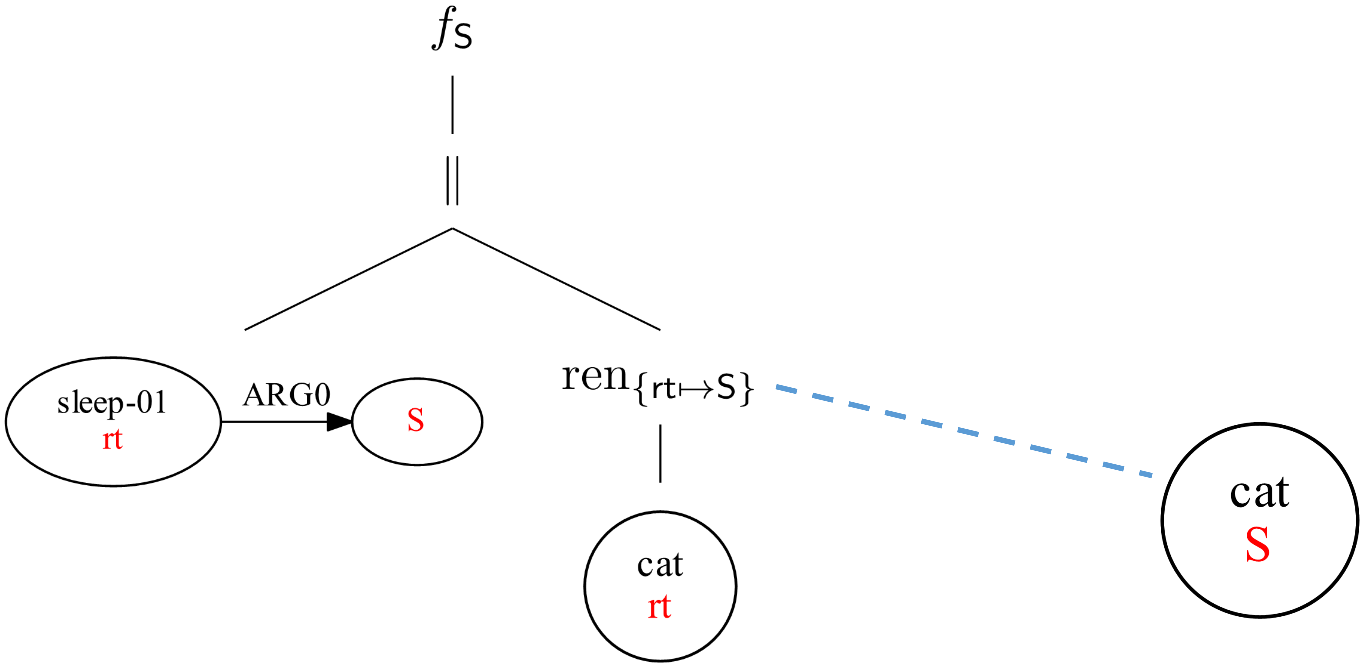
HR algebra (Courcelle, 1993)

Example:



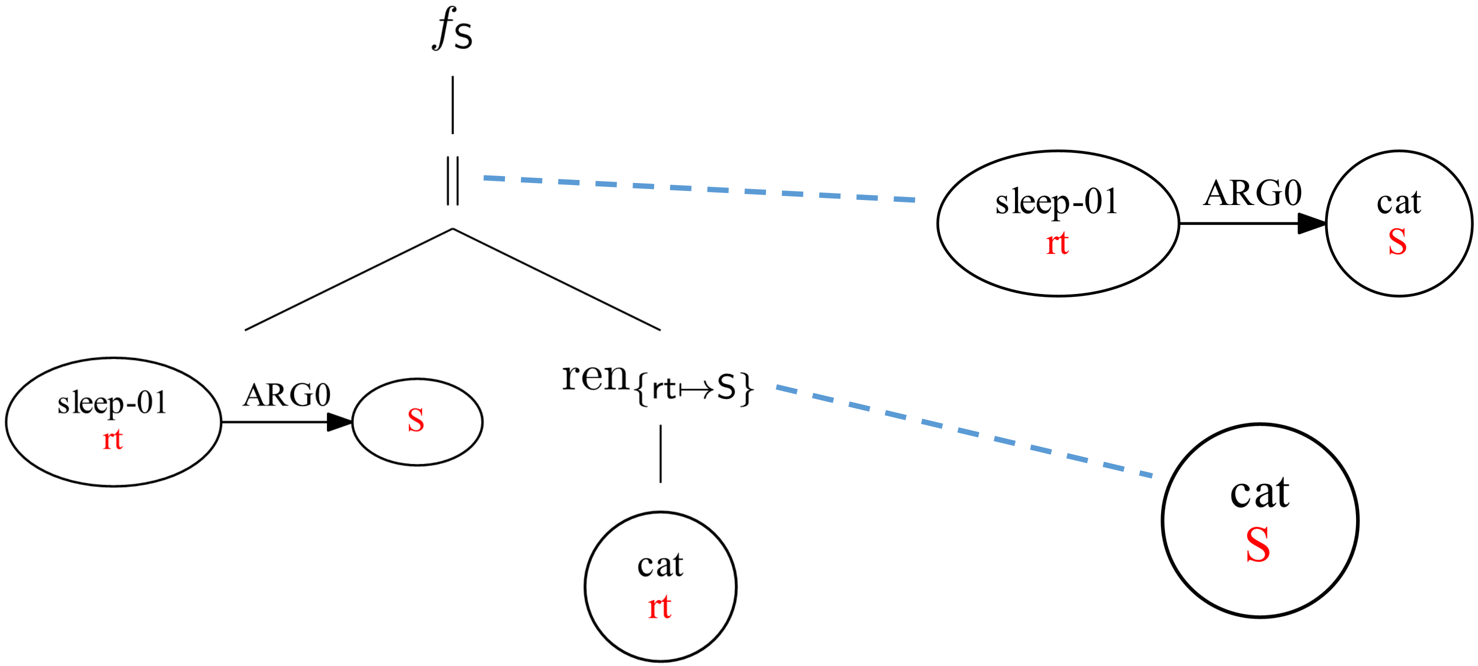
HR algebra (Courcelle, 1993)

Example:



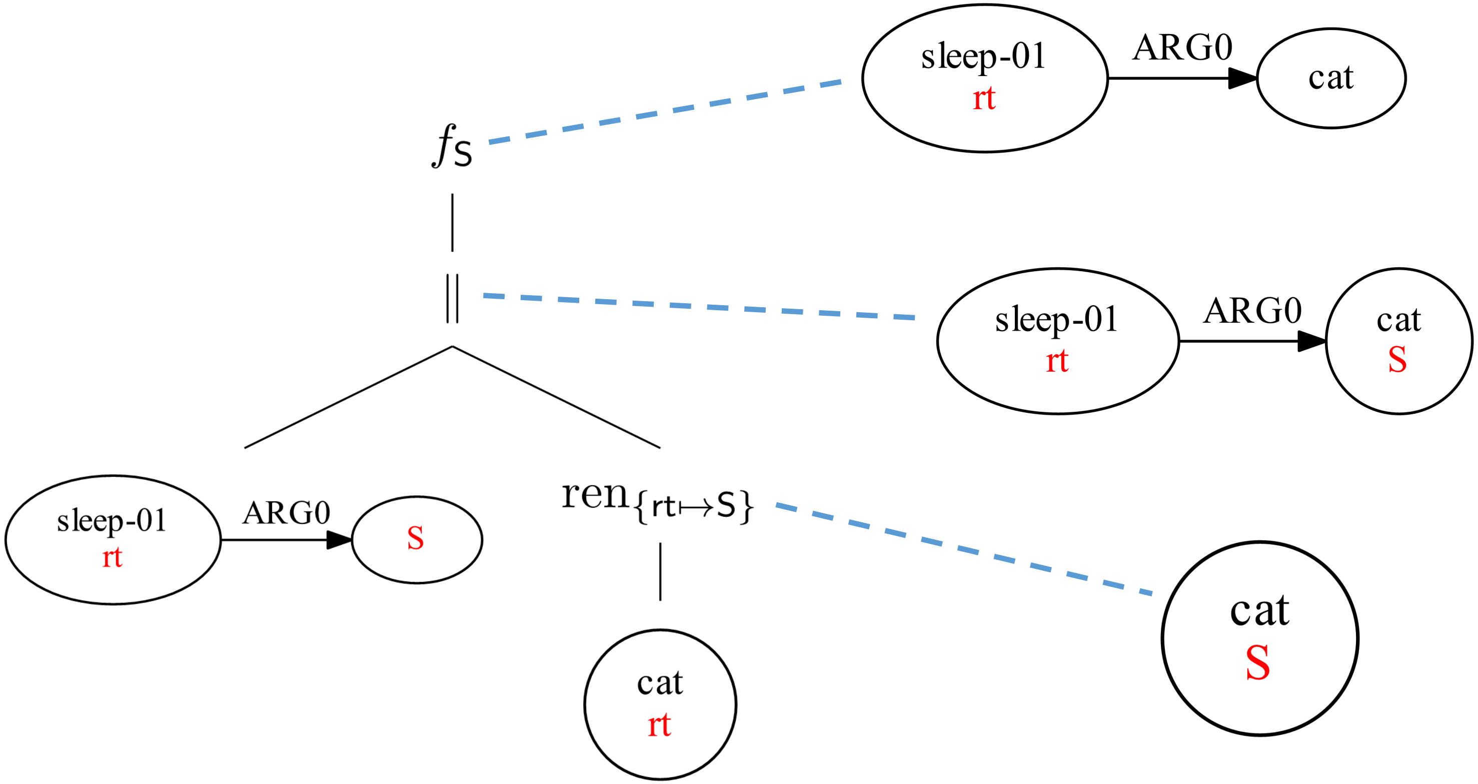
HR algebra (Courcelle, 1993)

Example:



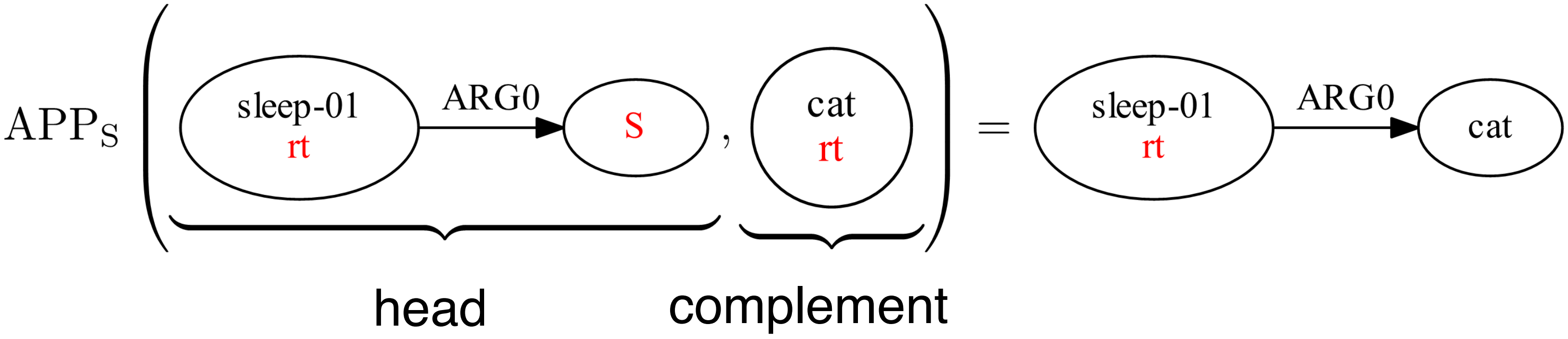
HR algebra (Courcelle, 1993)

Example:

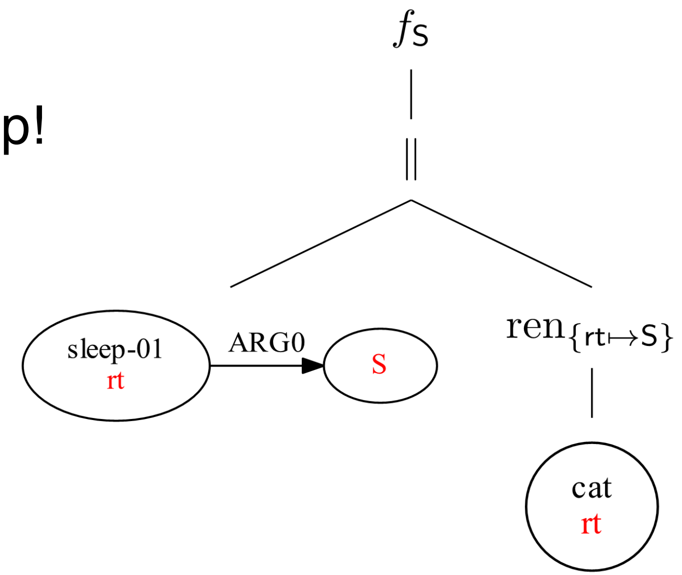


Apply operation

- Combines a **head** with a **complement**.

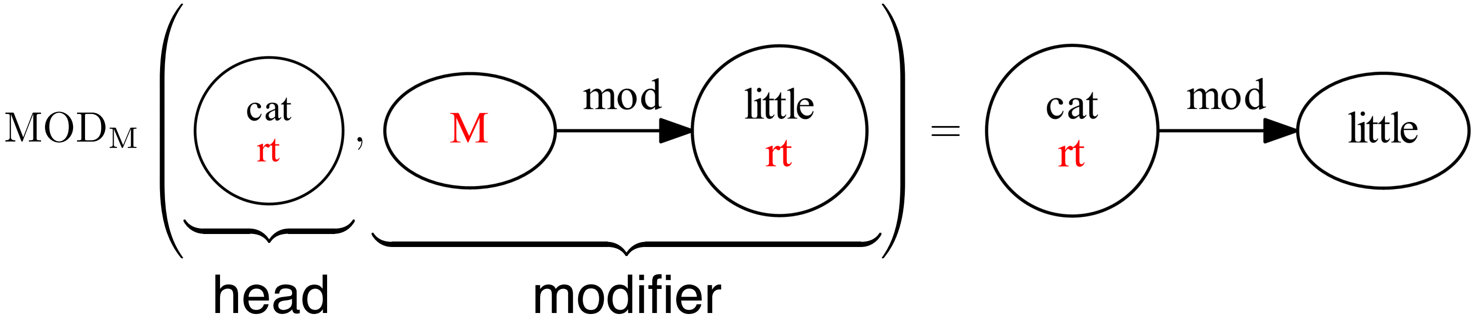


- Rename, merge and forget in one step!



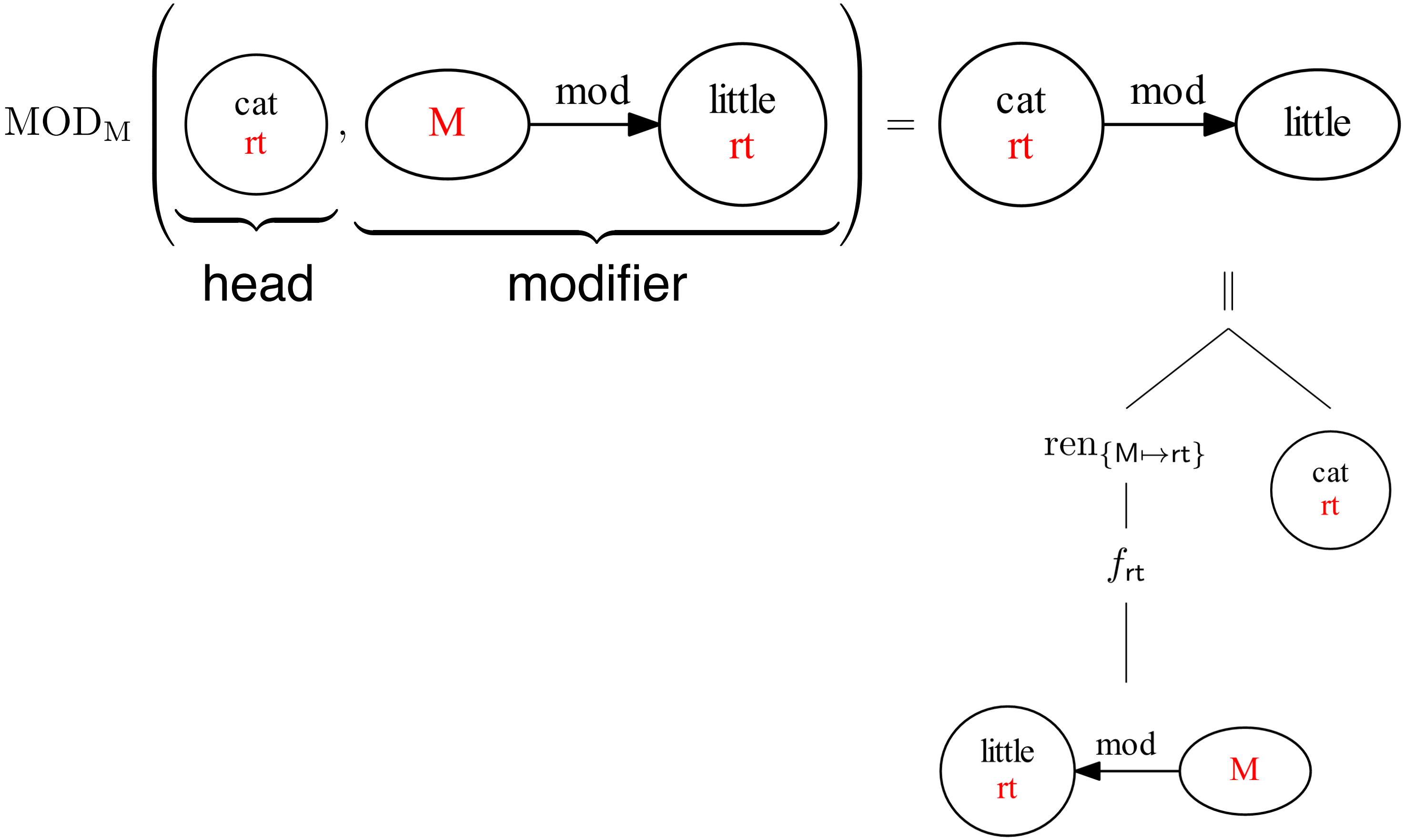
Modify operation

- Combines a **modifier** with a **head**.

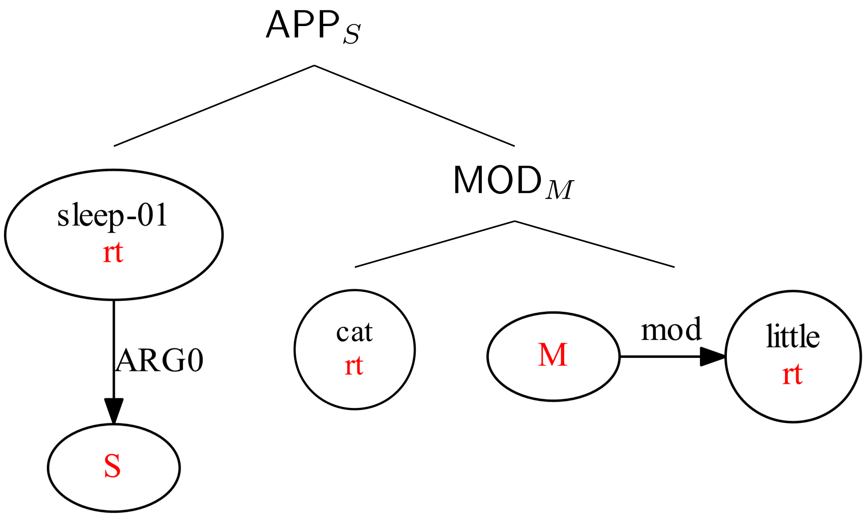


Modify operation

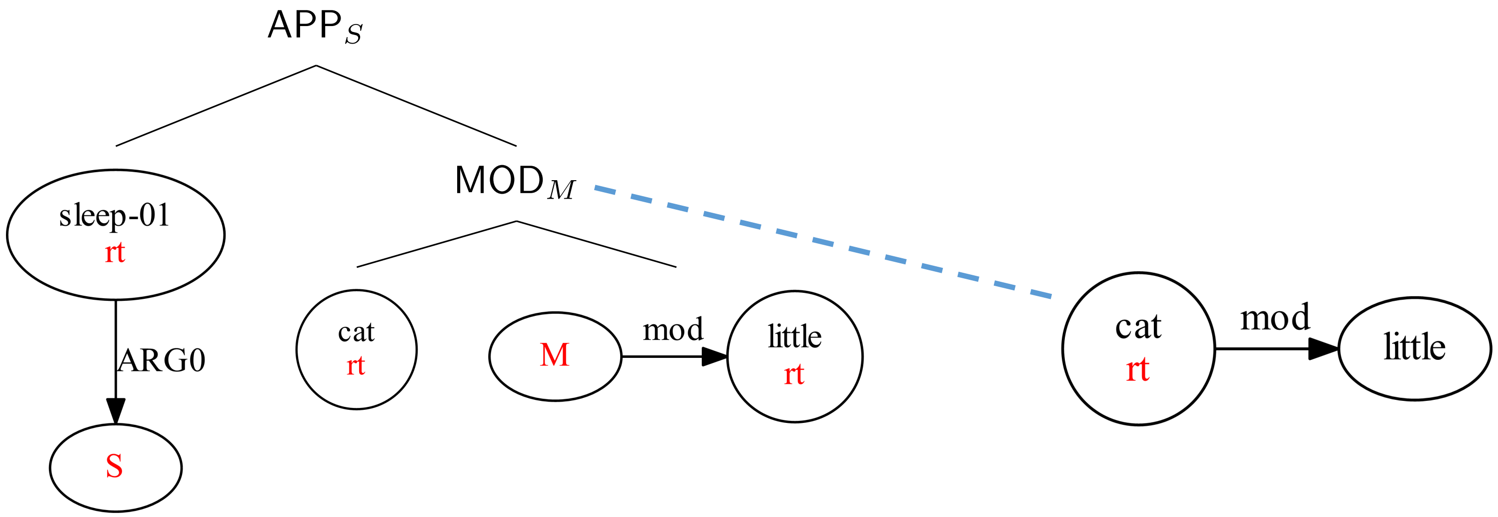
- Combines a **modifier** with a **head**.



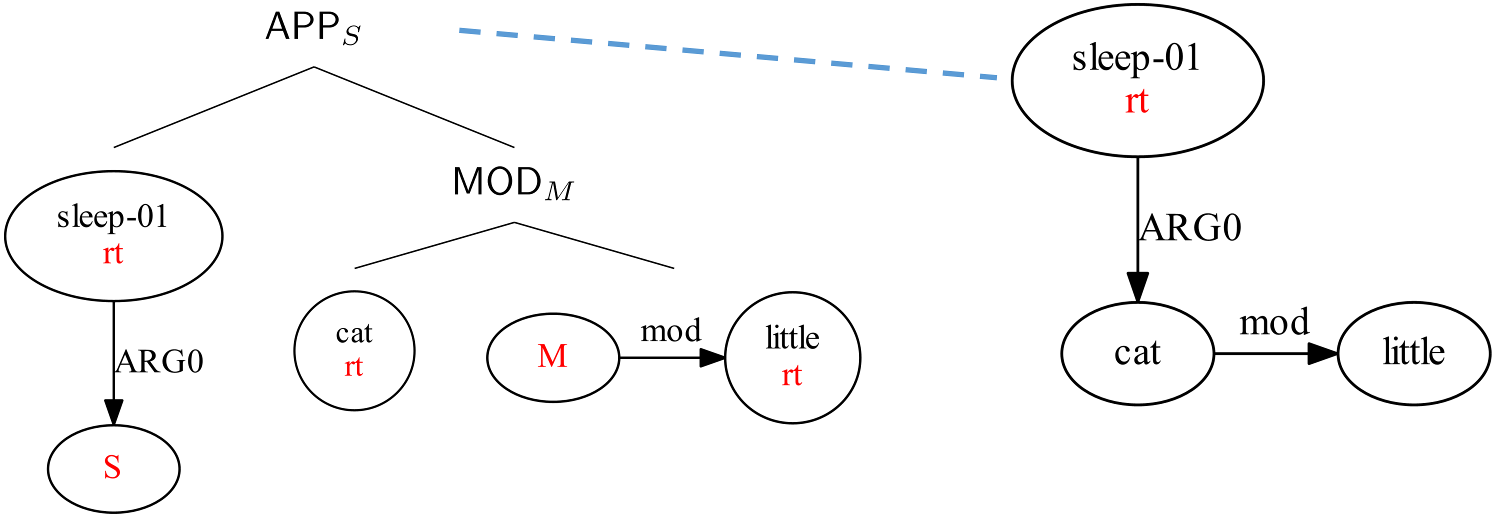
Example



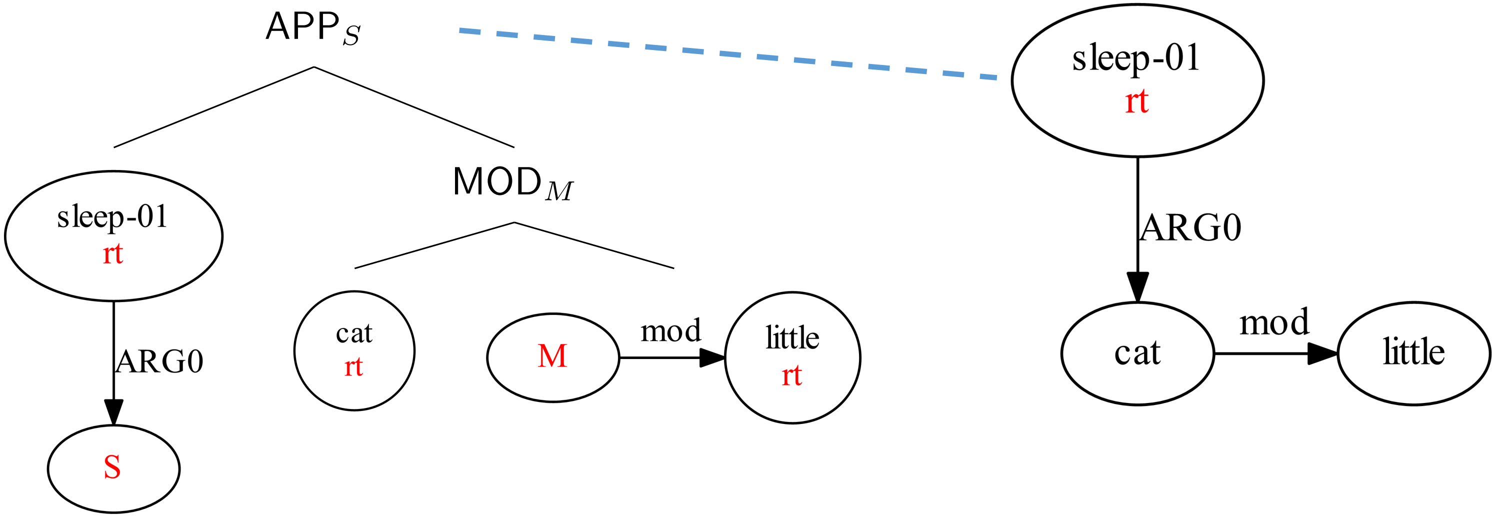
Example


APP$_S$
MOD$_M$
sleep-01
rt
ARG0
S
cat
rt
M
mod
little
rt
cat
rt
mod
little

Example

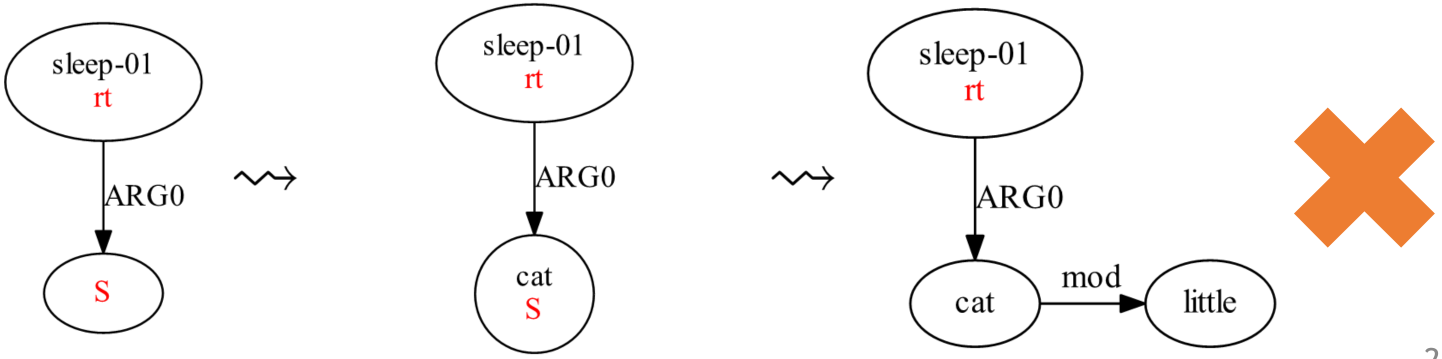

APP$_S$
sleep-01
rt
ARG0
S
MOD$_M$
cat
rt
M
mod
little
rt
sleep-01
rt
ARG0
cat
mod
little

Example



The complement must be “done” before we combine it.

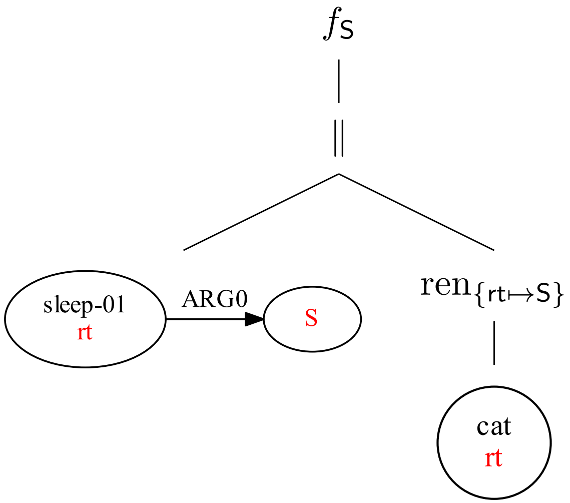
➢ This is the only term!



Linguistic Intuitions: APP

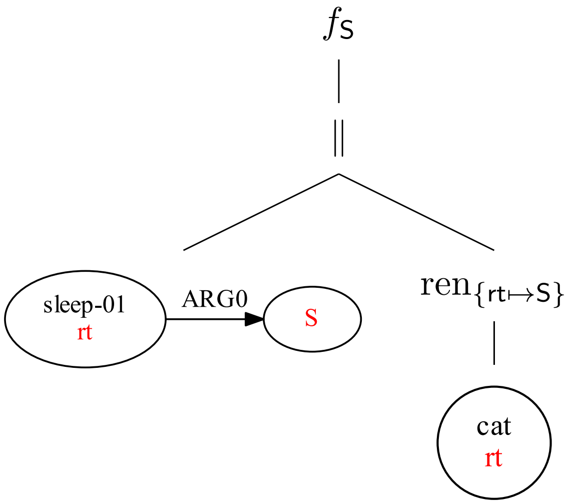
Linguistic Intuitions: APP

- Apply a function/head to its argument/complement



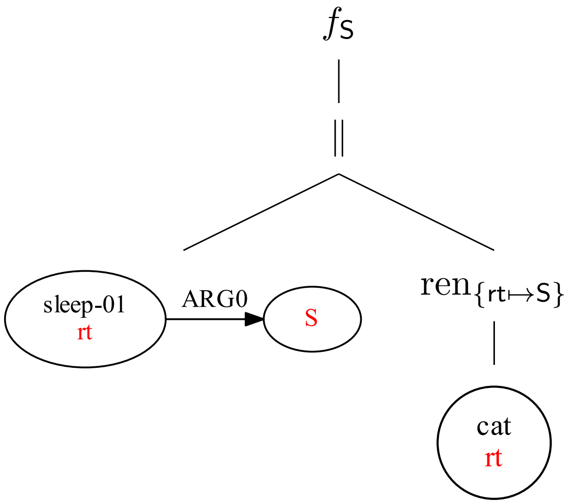
Linguistic Intuitions: APP

- Apply a function/head to its argument/complement
  - Two-way dependency:



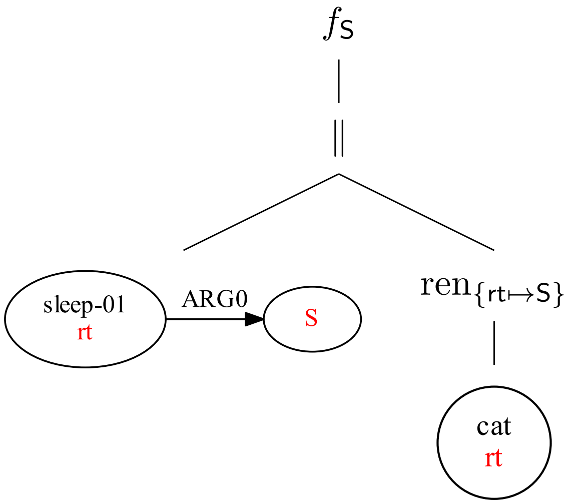
Linguistic Intuitions: APP

- Apply a function/head to its argument/complement
  - Two-way dependency:
    - argument needs to be selected



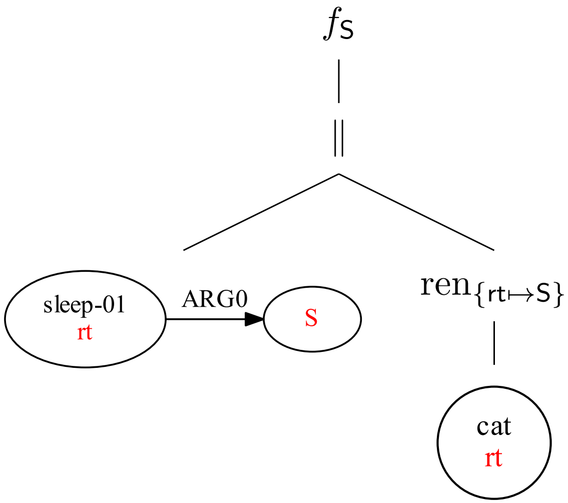
Linguistic Intuitions: APP

- Apply a function/head to its argument/complement
    - Two-way dependency:
        - argument needs to be selected
        - Function needs an argument



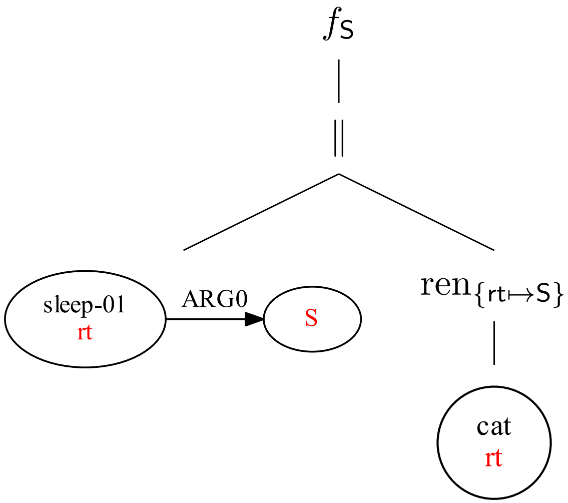
Linguistic Intuitions: APP

- Apply a function/head to its argument/complement
  - Two-way dependency:
    - argument needs to be selected
    - Function needs an argument
  - Unfilled argument slots marked with sources like S for subject

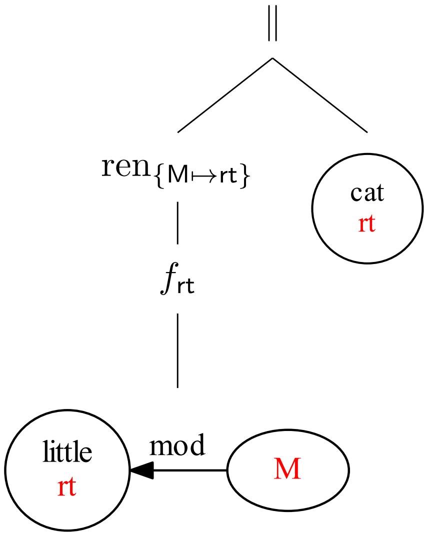


Linguistic Intuitions: APP

- Apply a function/head to its argument/complement
  - Two-way dependency:
    - argument needs to be selected
    - Function needs an argument
  - Unfilled argument slots marked with sources like S for subject
    - Syntax: like theta grids in lexical entries

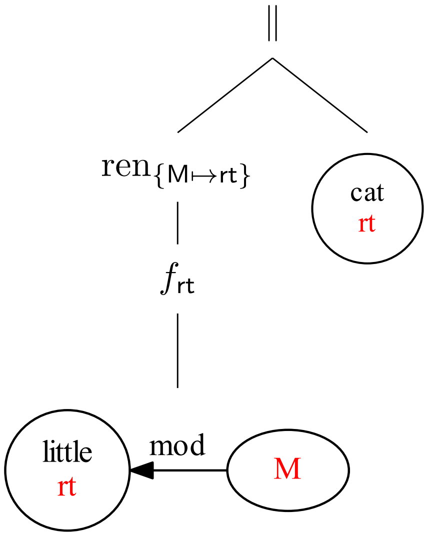


Linguistic Intuitions: MOD



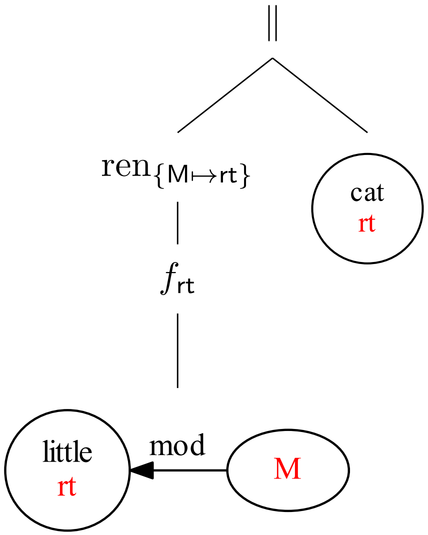
Linguistic Intuitions: MOD

- Add a modifier to a head



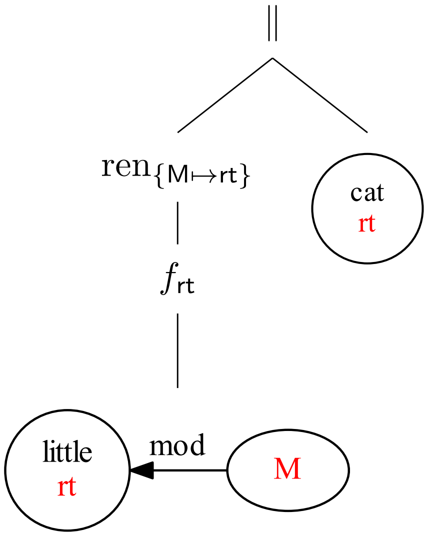
Linguistic Intuitions: MOD

- Add a modifier to a head
    - One-way dependency:



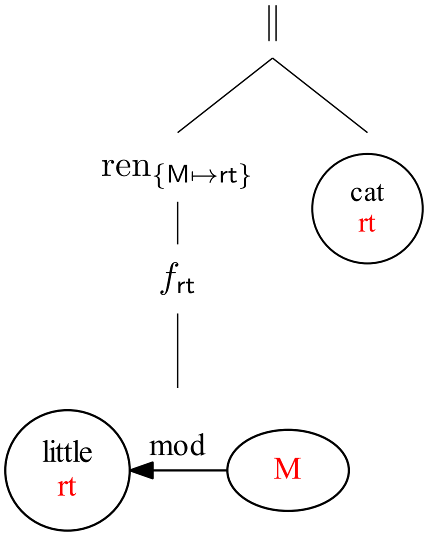
Linguistic Intuitions: MOD

- Add a modifier to a head
    - One-way dependency:
        - Modifier needs to modify



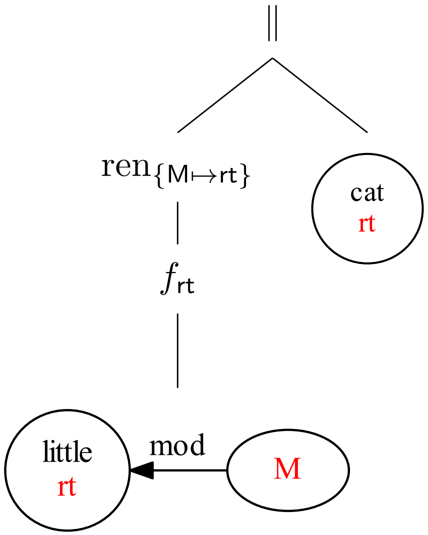
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn't need to be modified



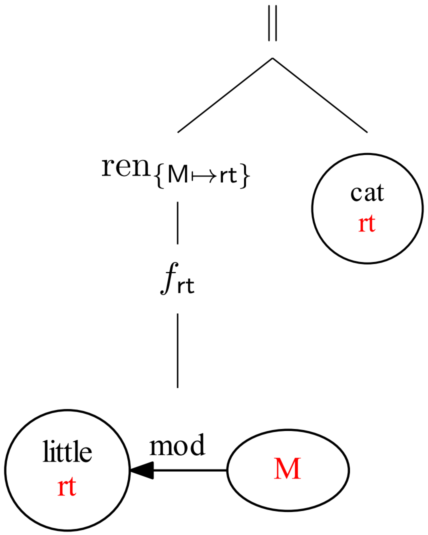
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn't need to be modified
  - Always modify at the root:



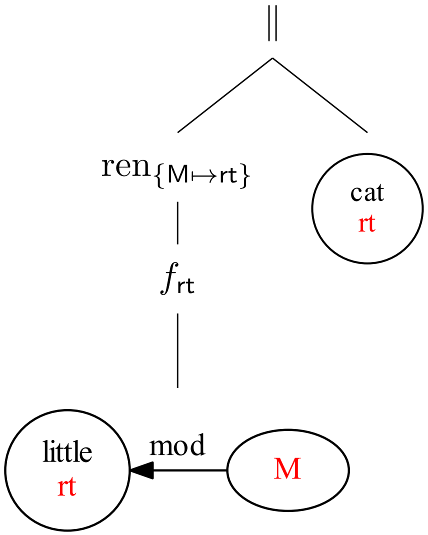
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn't need to be modified
  - Always modify at the root:
    - Modifier isn't filling a need in the head, just adding itself in



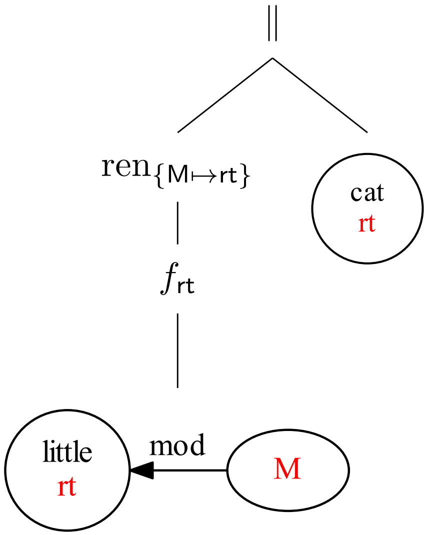
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn't need to be modified
  - Always modify at the root:
    - Modifier isn't filling a need in the head, just adding itself in
    - Syntax: modification is adjunction of two complete phrases, yielding a phrase of the same kind we started with, eg:



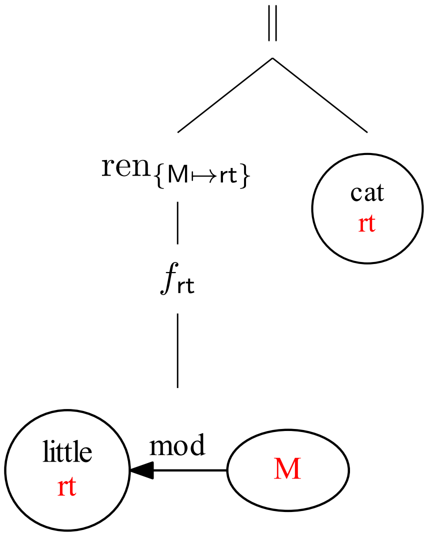
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn’t need to be modified
  - Always modify at the root:
    - Modifier isn’t filling a need in the head, just adding itself in
    - Syntax: modification is adjunction of two complete phrases, yielding a phrase of the same kind we started with, eg:
      - NP -> AP NP



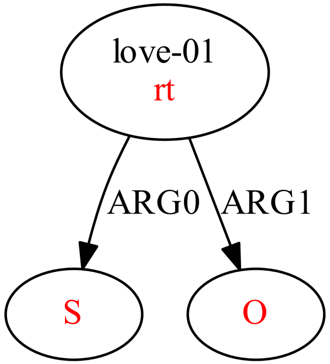
Linguistic Intuitions: MOD

- Add a modifier to a head
  - One-way dependency:
    - Modifier needs to modify
    - Head doesn't need to be modified
  - Always modify at the root:
    - Modifier isn't filling a need in the head, just adding itself in
    - Syntax: modification is adjunction of two complete phrases, yielding a phrase of the same kind we started with, eg:
      - NP -> AP NP
    - Optional: *type* is unchanged

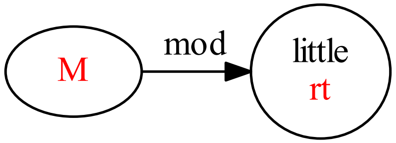


Graph types

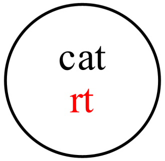
Type of an s- graph (take one): the set of its **non-rt source names**.



has type [S,O]



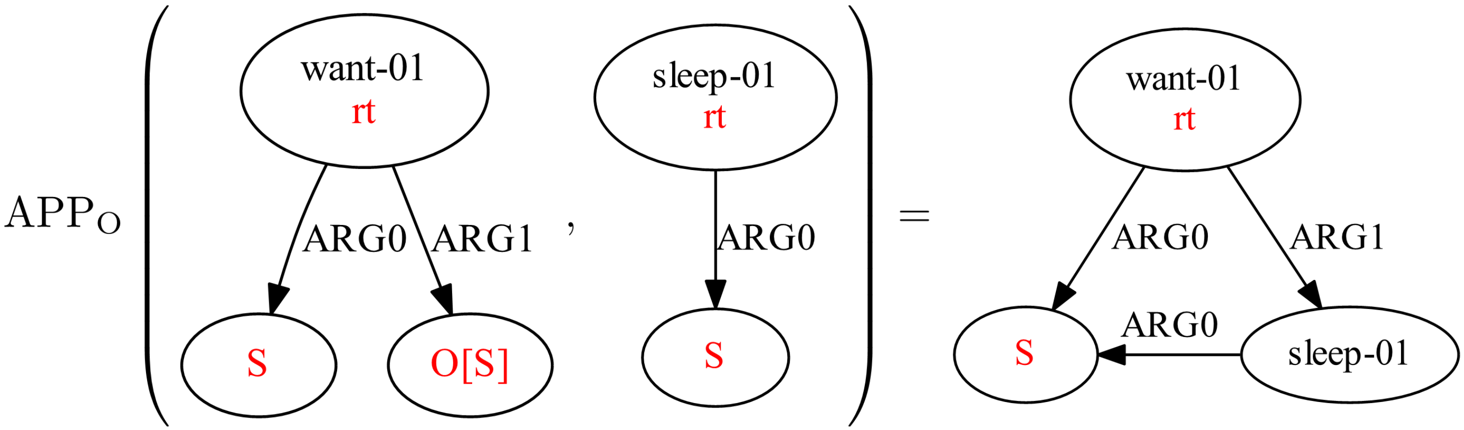
has type [M]



has empty type

Source name annotations

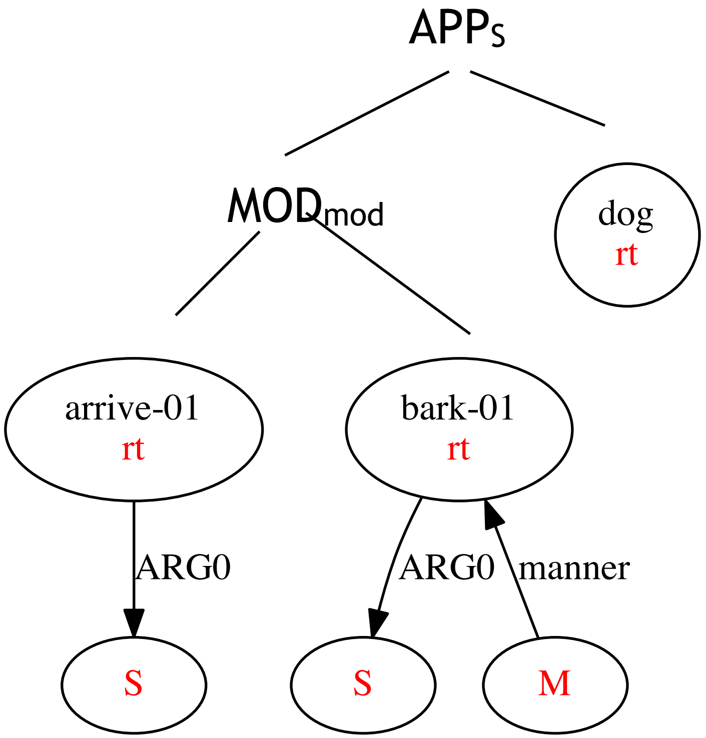
- O[S] is an O source **annotated** with type [S].
- It means that we require the O-complement to have type [S].

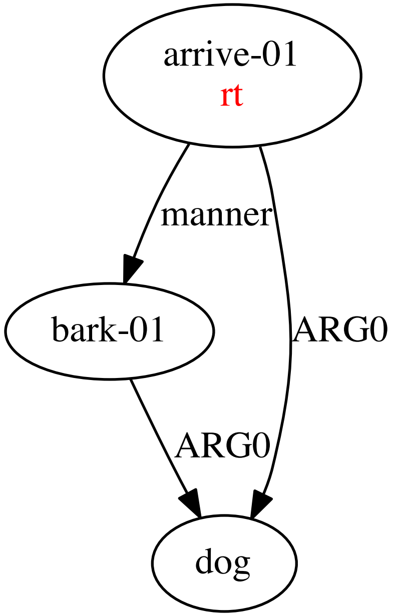


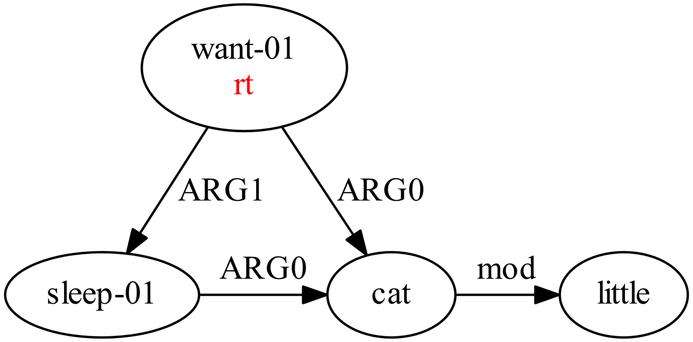
- At other sources, we require the complement to have the empty type (only rt-source).

MOD type requirement

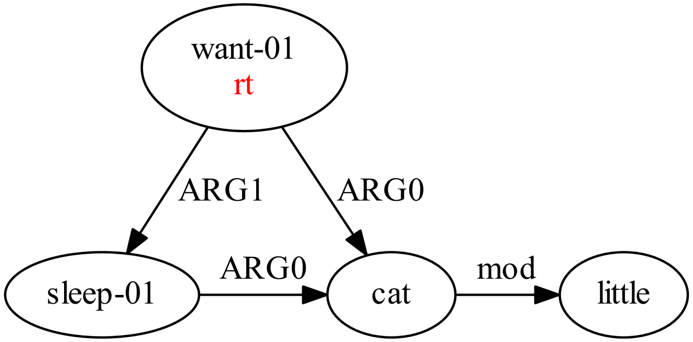
- MOD_{mod} allowed iff type of modifier, minus mod, is a subset of the head's type
    - —> Type of result is the same as the type of the head



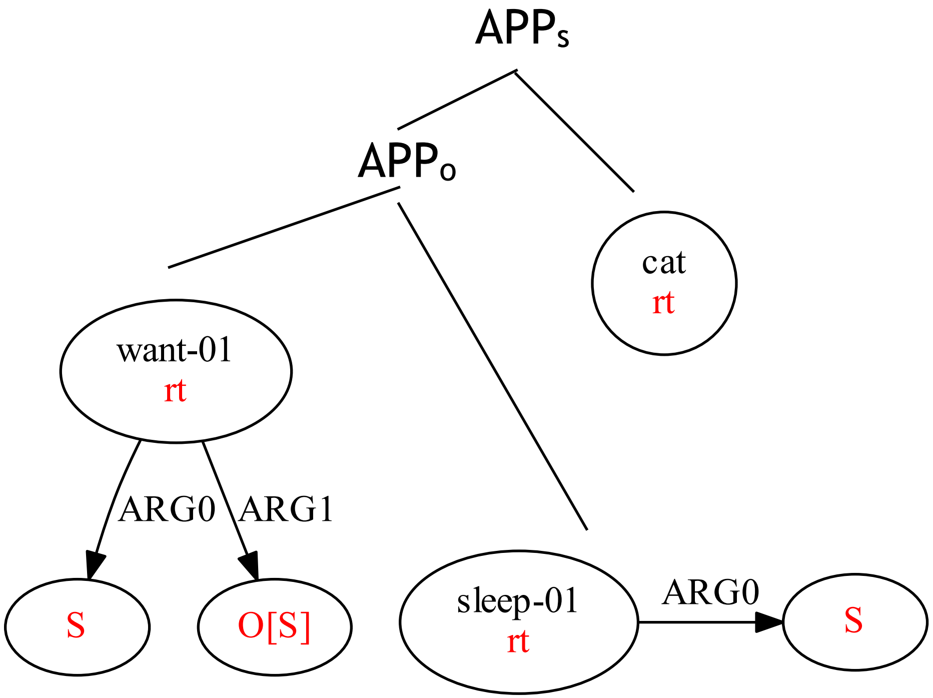




“The little cat wants to sleep.”



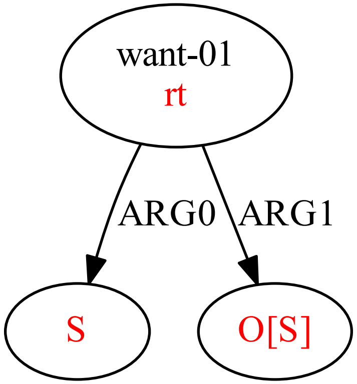
”The little cat wants to sleep.”



Order of operations

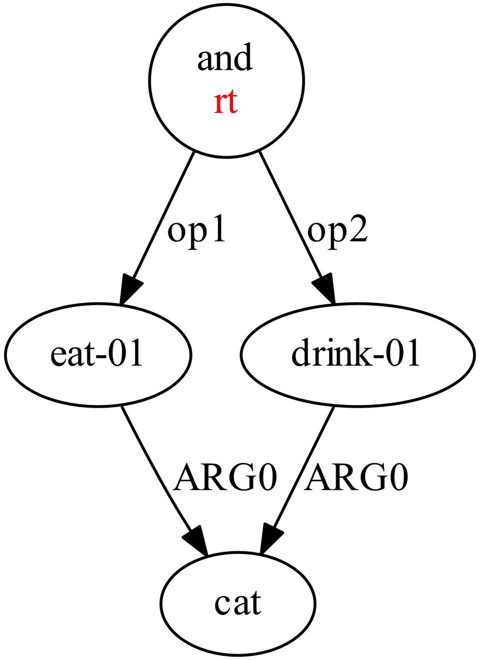
- Restriction: if an annotation introduces a source that is present in the graph, the annotated source must be filled first

- e.g.: APP_o before APP_s in subject control



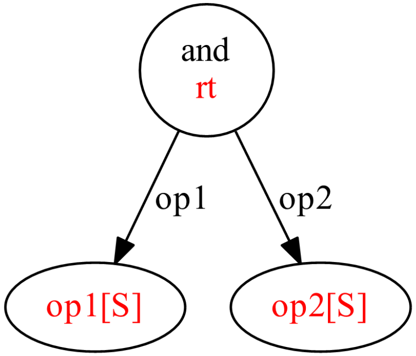
- We're not totally sure what the status of this restriction should be

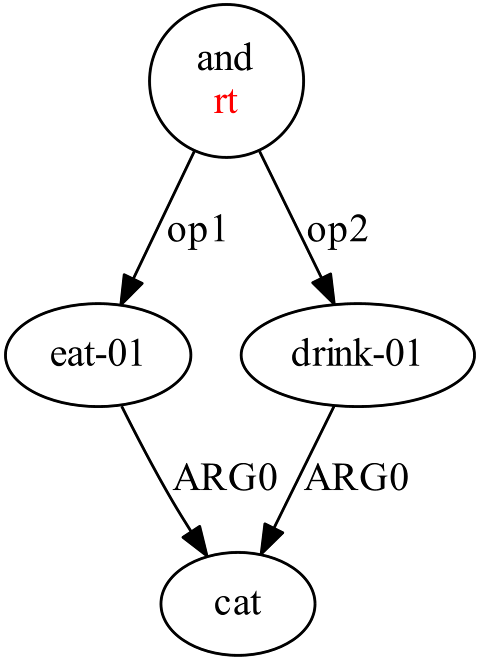
Coordination



“The cat eats and drinks.”

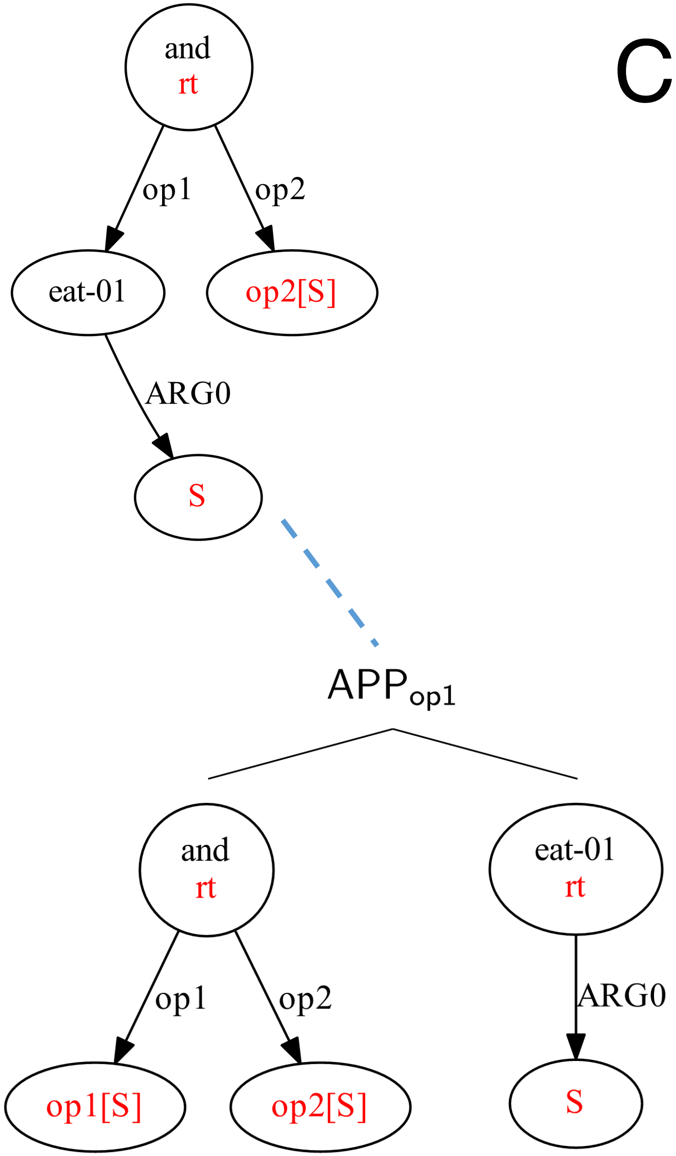
Coordination

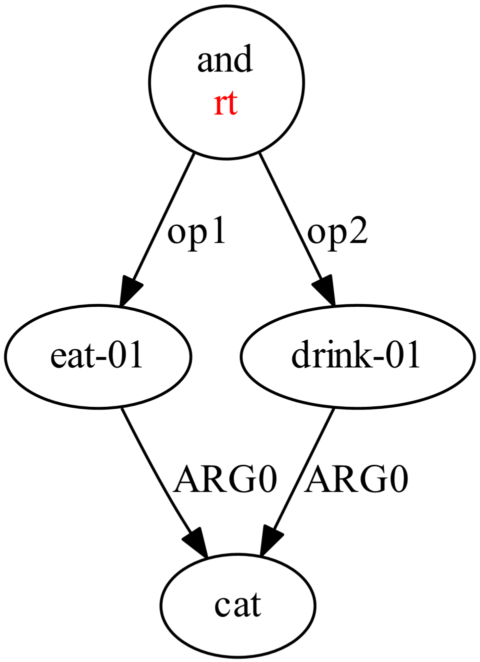




“The cat eats and drinks.”

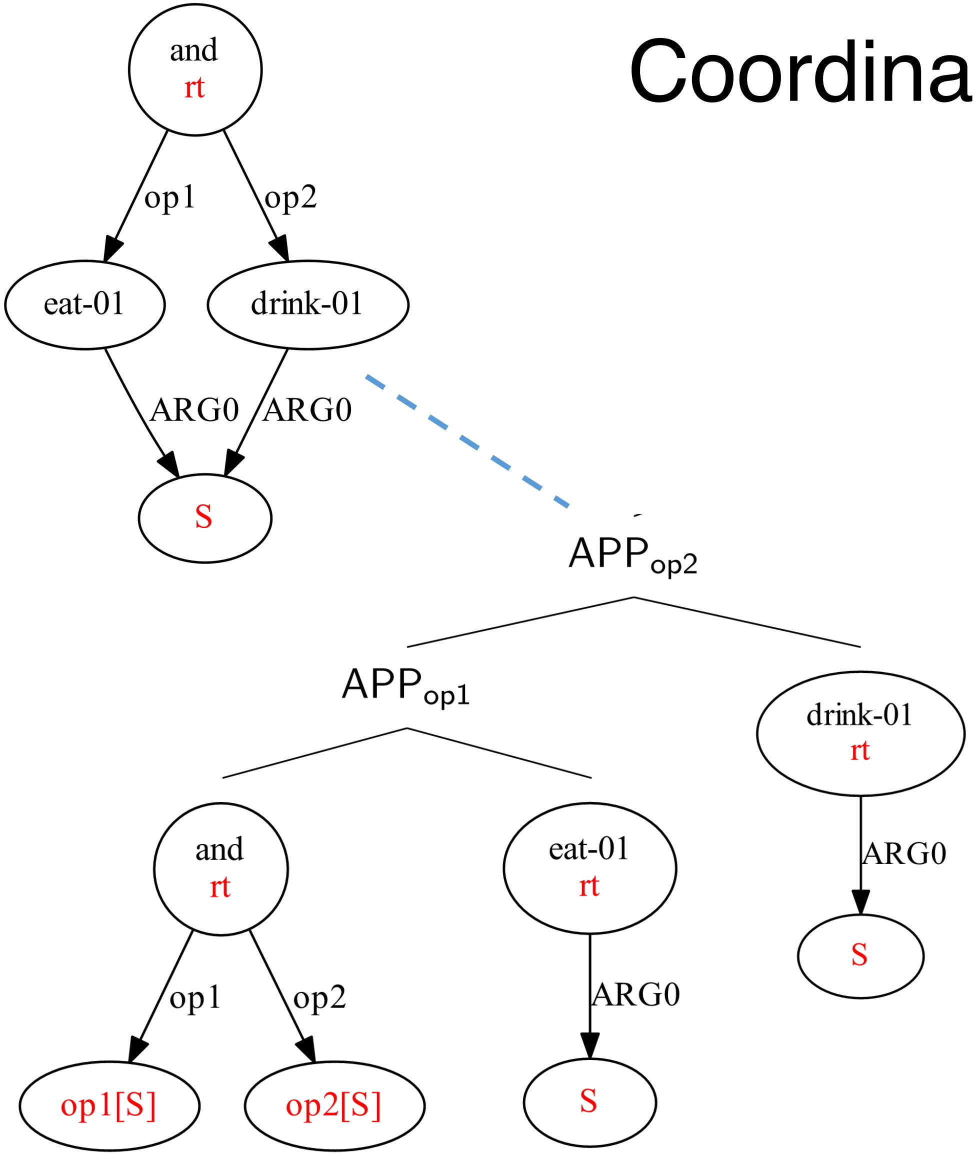
Coordination

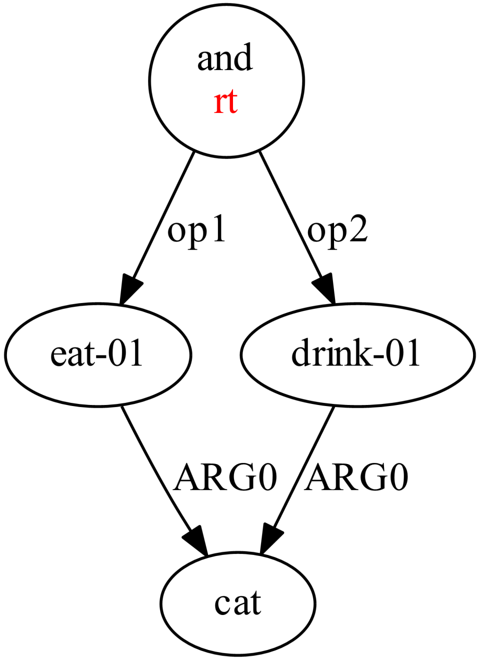




“The cat eats and drinks.”

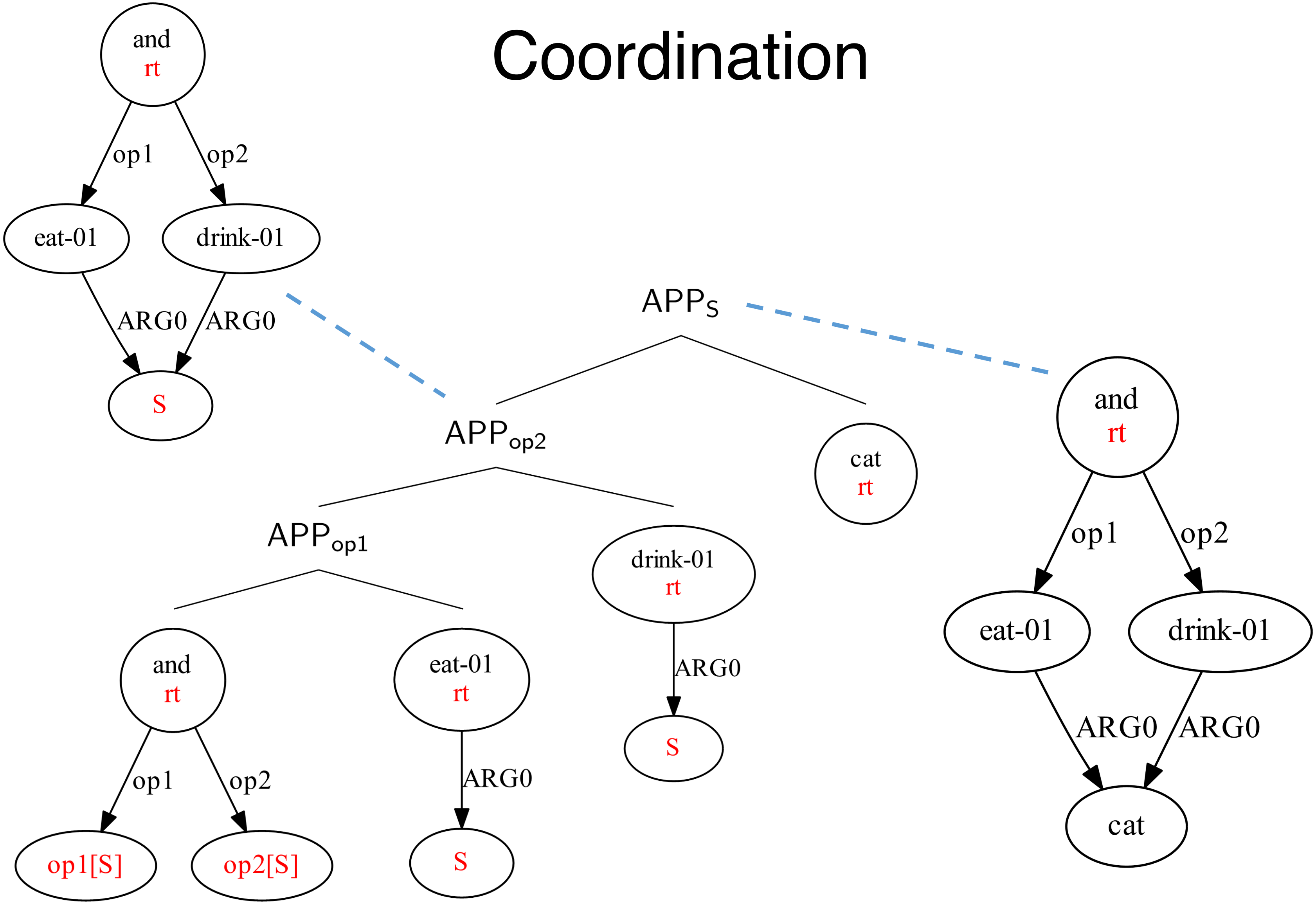
Coordination





“The cat eats and drinks.”

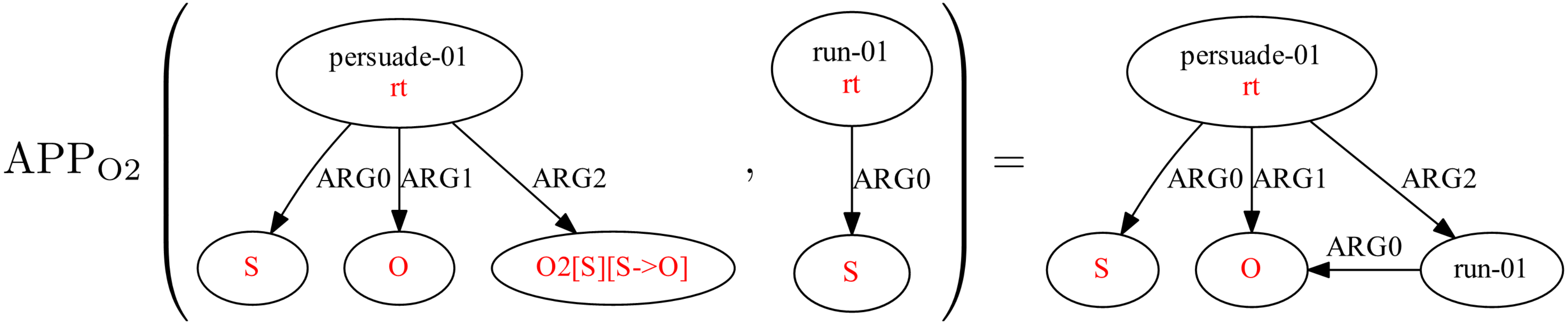
Coordination

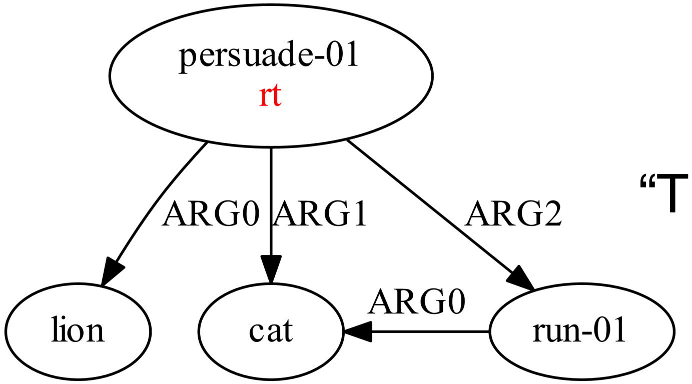


“The cat eats and drinks.”

Object control

- [S->O] adds another rename to APP_{O2}

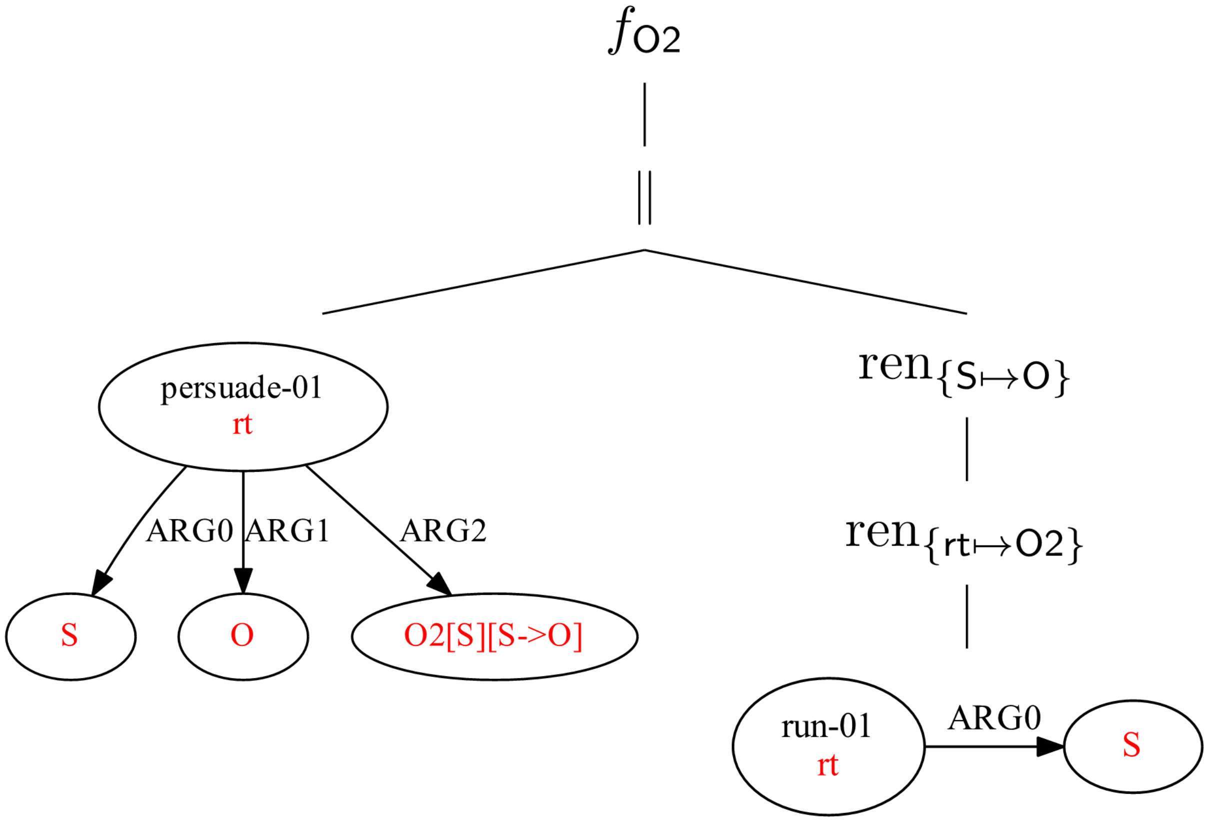




“The lion persuaded the cat to run.”

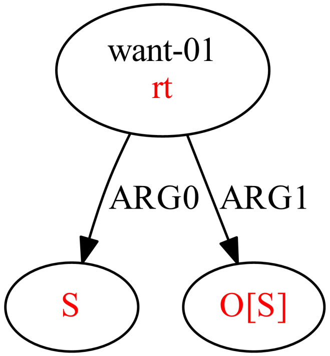
Object control

- corresponding HR term:

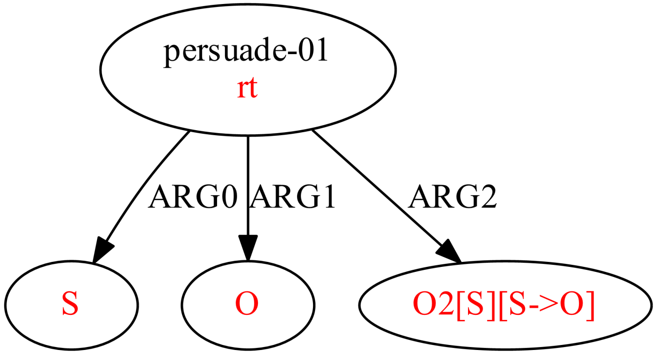

f_{O2}
$||$
persuade-01
rt
ARG0
ARG1
ARG2
S
O
O2[S][S->O]
$\mathrm{ren}_{\{\mathsf{S} \mapsto \mathsf{O}\}}$
$\mathrm{ren}_{\{\mathsf{rt} \mapsto \mathsf{O2}\}}$
run-01
rt
ARG0
S

Graph types (update)

Type of a graph: the set of its non-rt source names, **and their annotations**.

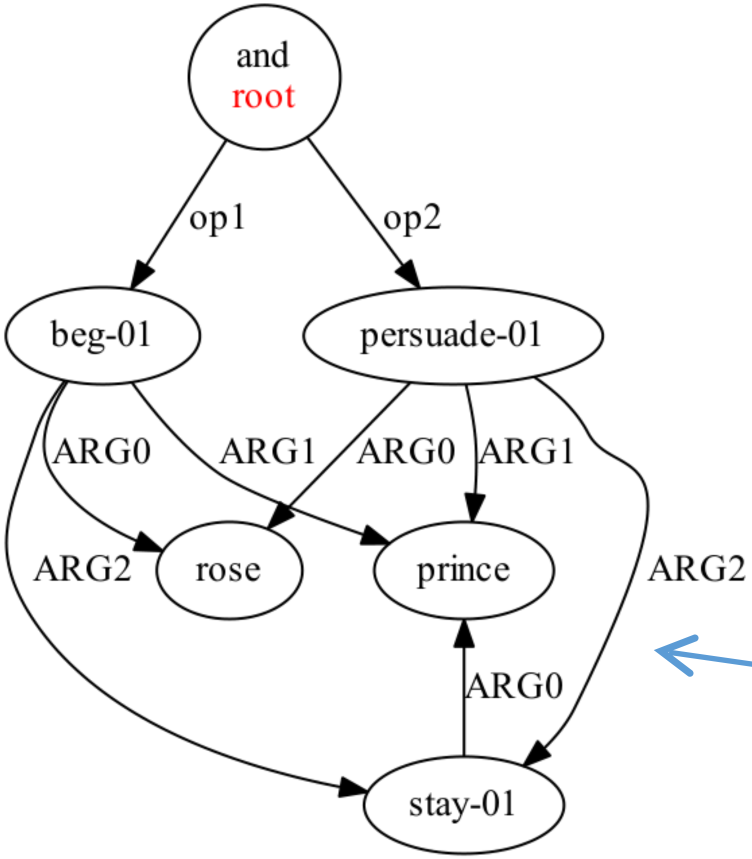


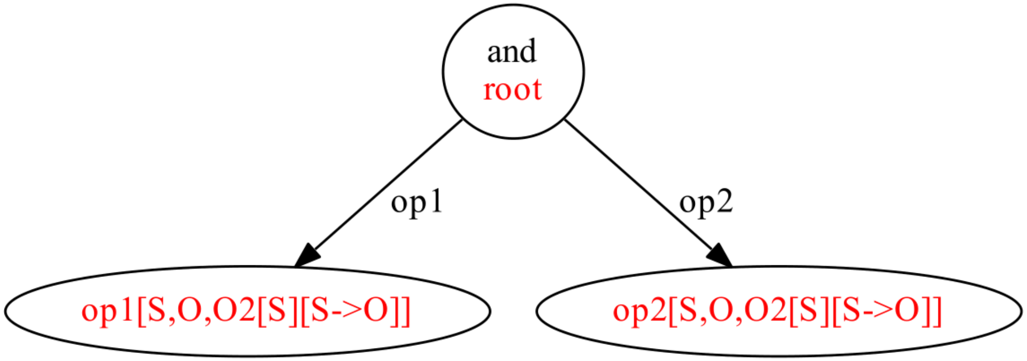
has type [S,O[S]]

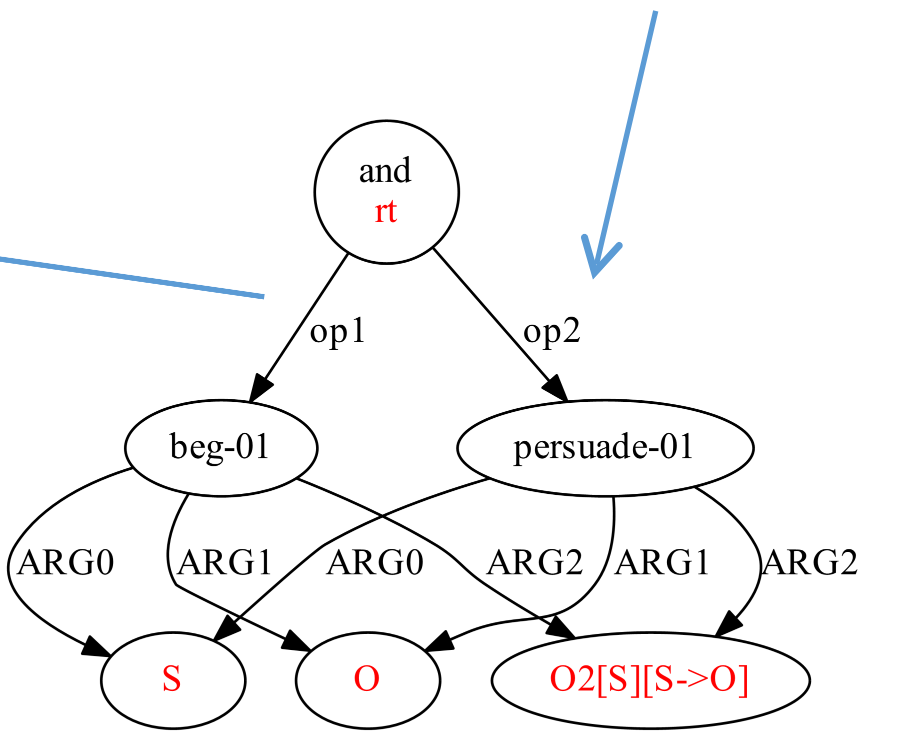


has type [S,O,O2[S][S->O]]

Coordination of control verbs



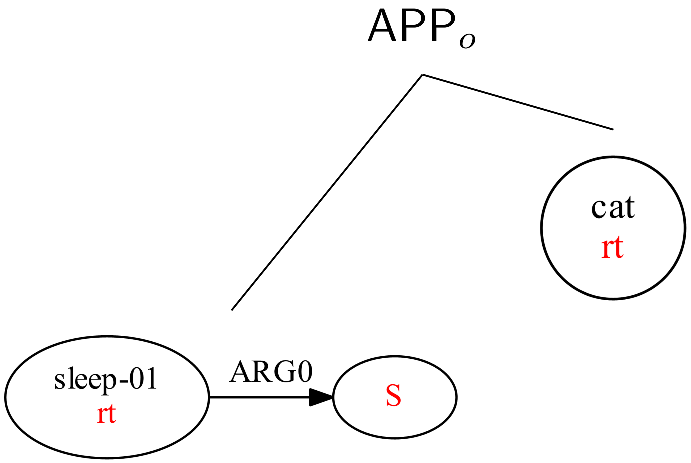




“The rose begged and persuaded the prince to stay.”

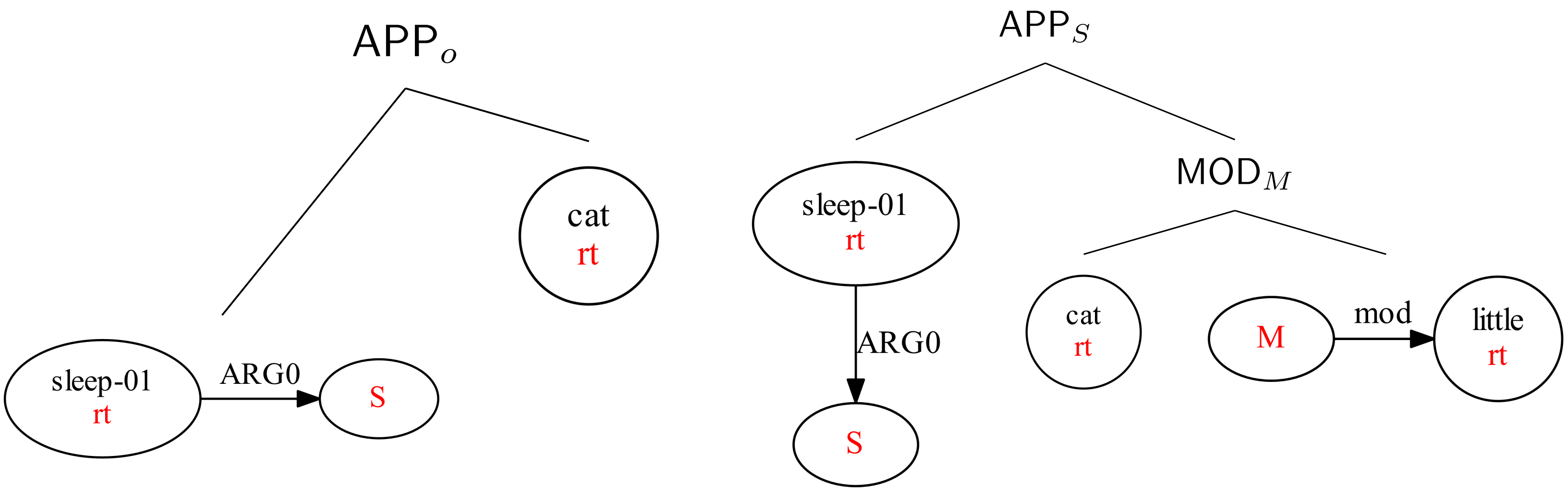
Well-typed term

An AM-term is *well-typed* iff it evaluates to an AS-graph with the empty type.

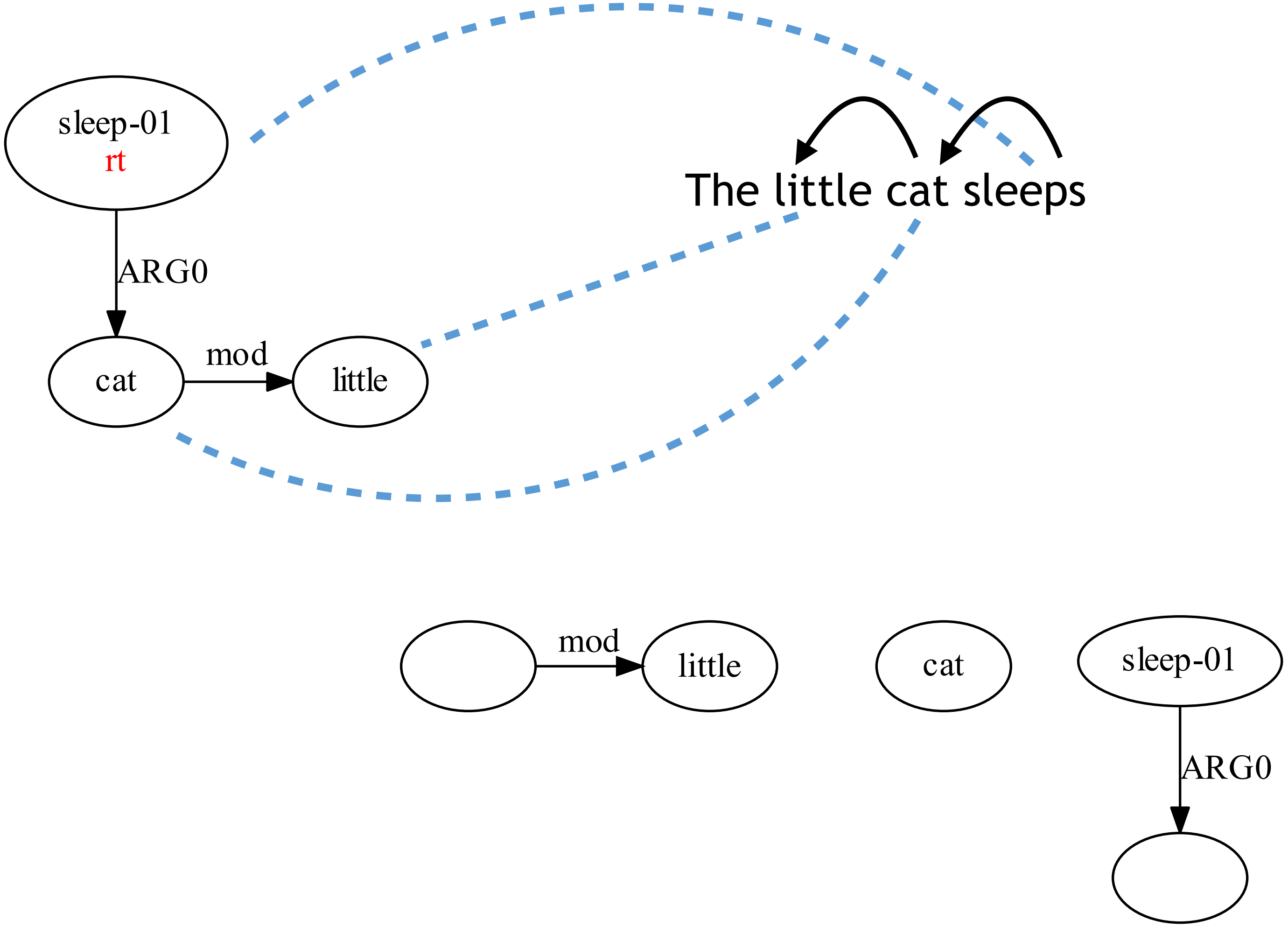


Well-typed term

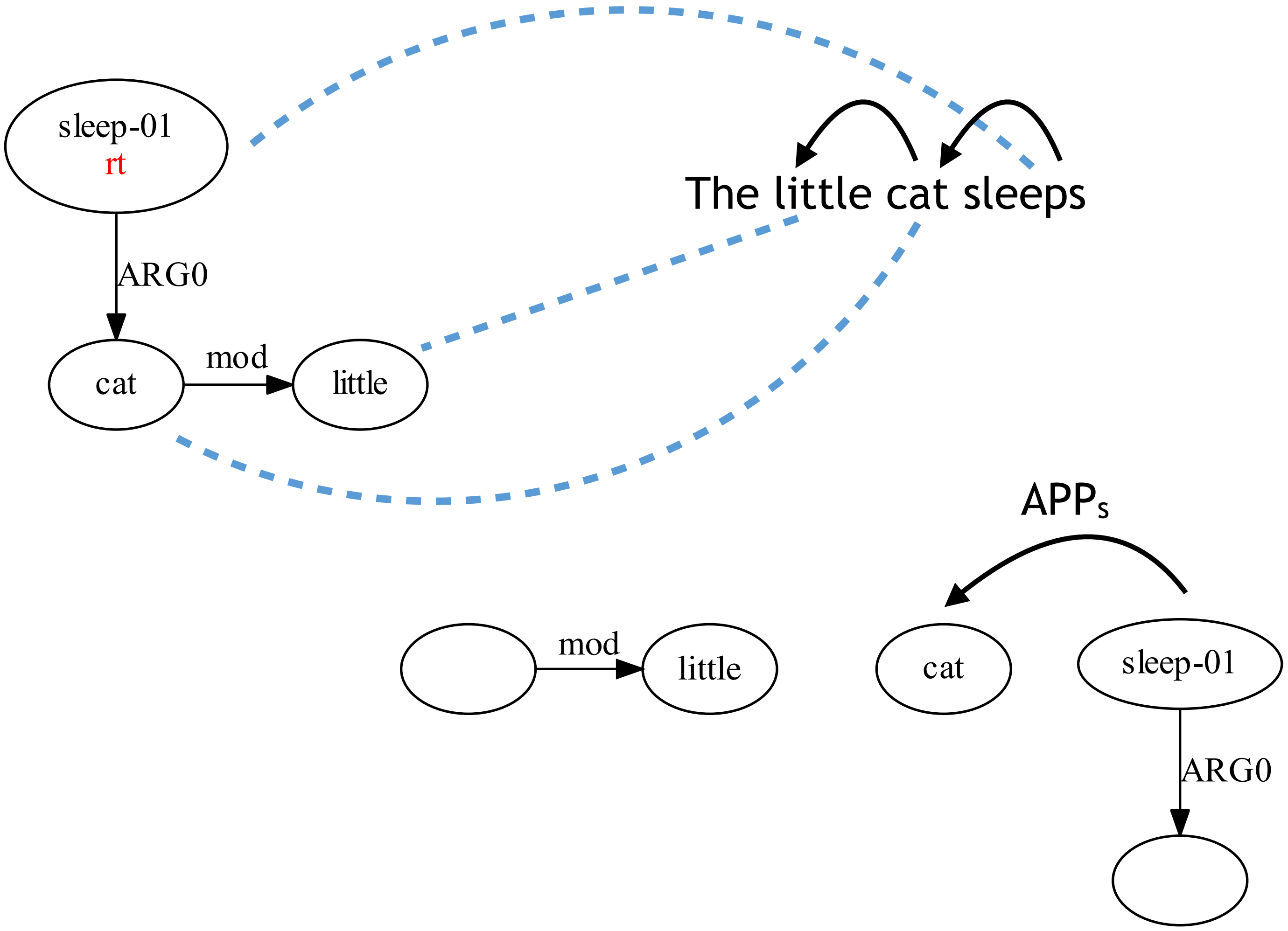
An AM-term is *well-typed* iff it evaluates to an AS-graph with the empty type.



Dependency Parsing into AMRs

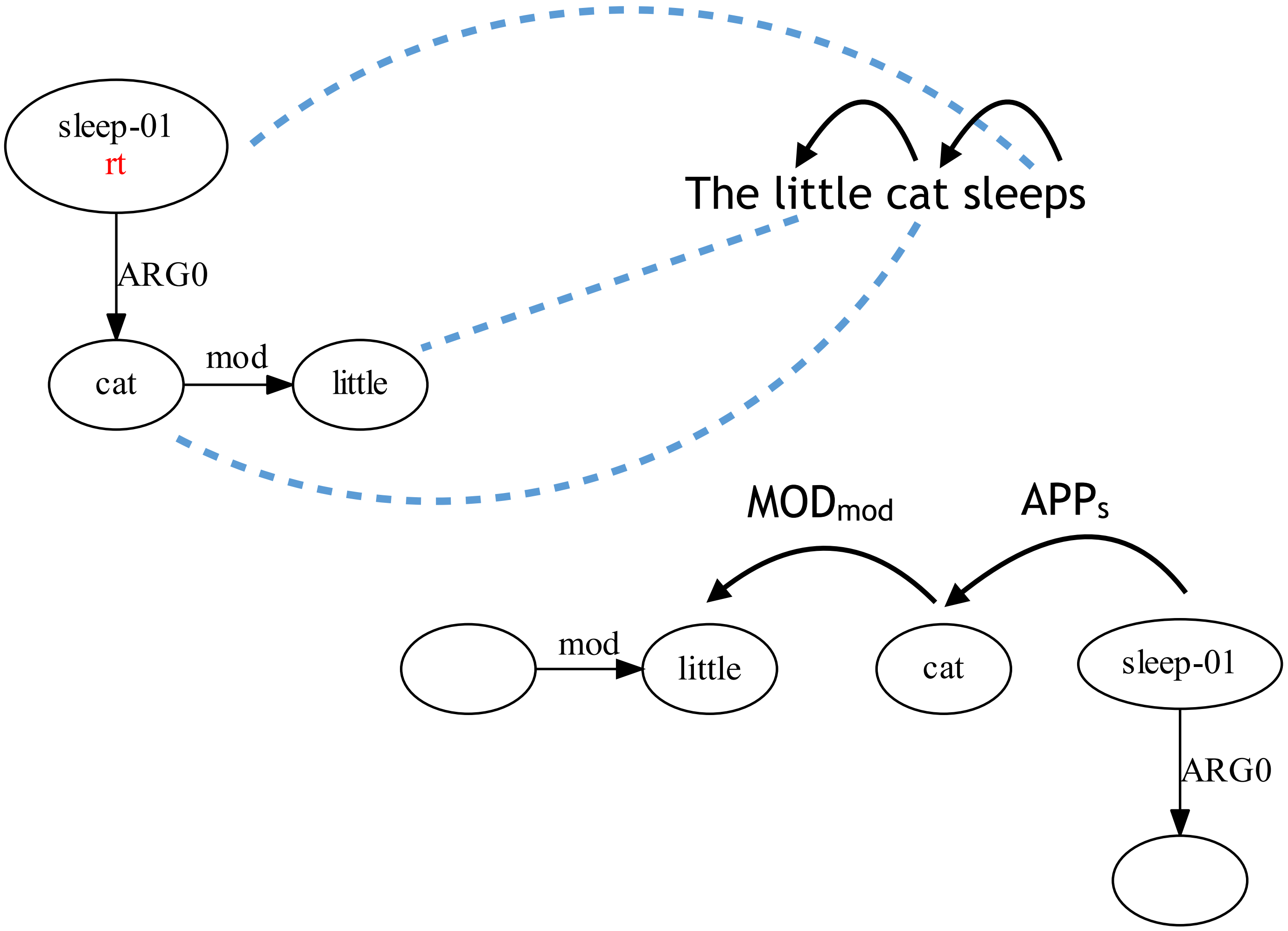


Dependency Parsing into AMRs



The little cat sleeps

Dependency Parsing into AMRs



Conclusion

- Graphs have a hidden compositional structure brought out by the AM algebra
- Lexicalised semantic dependencies
- Syntactic alternations lexicalised in source choices