# Sequence-to-sequence Models for Cache Transition Systems

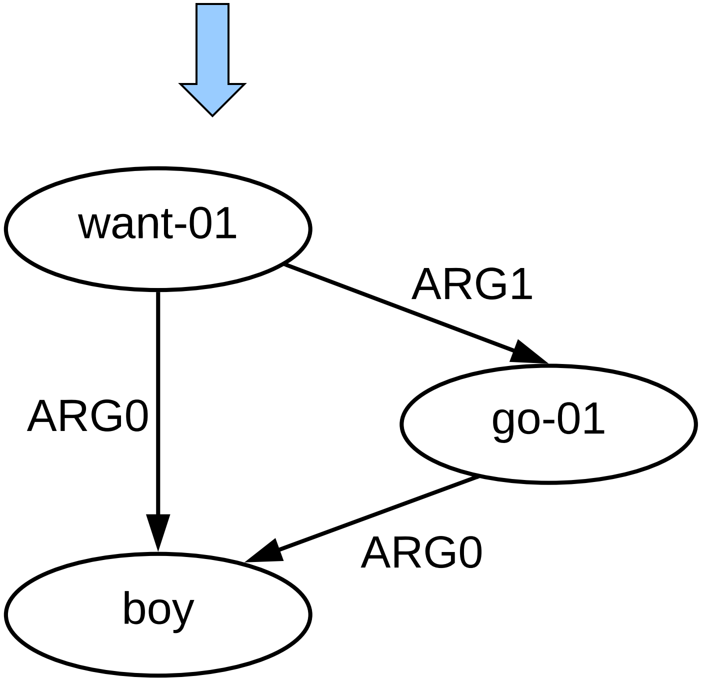
Xiaochang Peng[1], **Linfeng Song**[1], Daniel Gildea[1] and Giorgio Satta[2]

1 UNIVERSITY of ROCHESTER

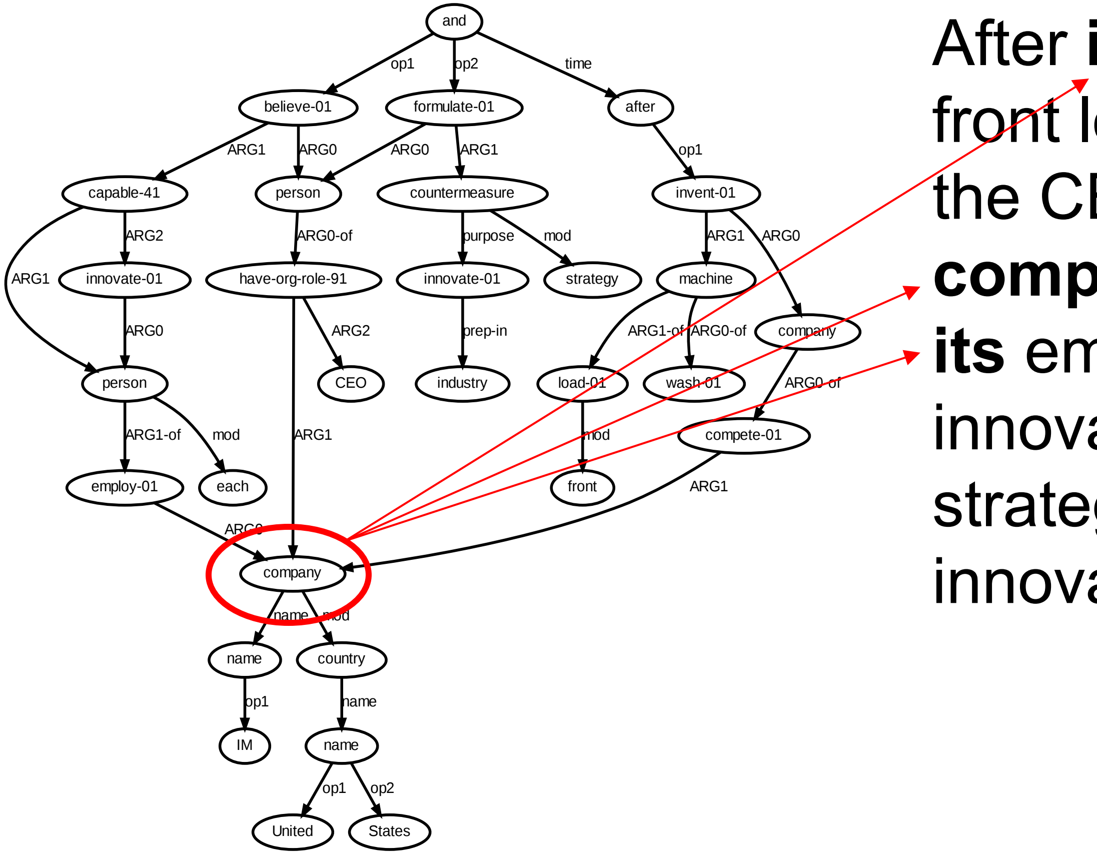
2 UNIVERSITÀ DEGLI STUDI DI PADOVA

# AMR

- “John wants to go”



# AMR

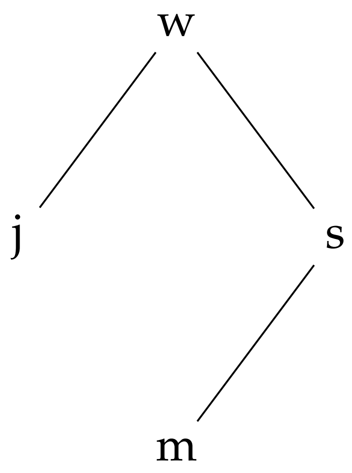


After **its** competitor invented the front loading washing machine, the CEO of the American **IM company** believed that each of **its** employees had the ability for innovation, and formulated strategic countermeasures for innovation in the industry.

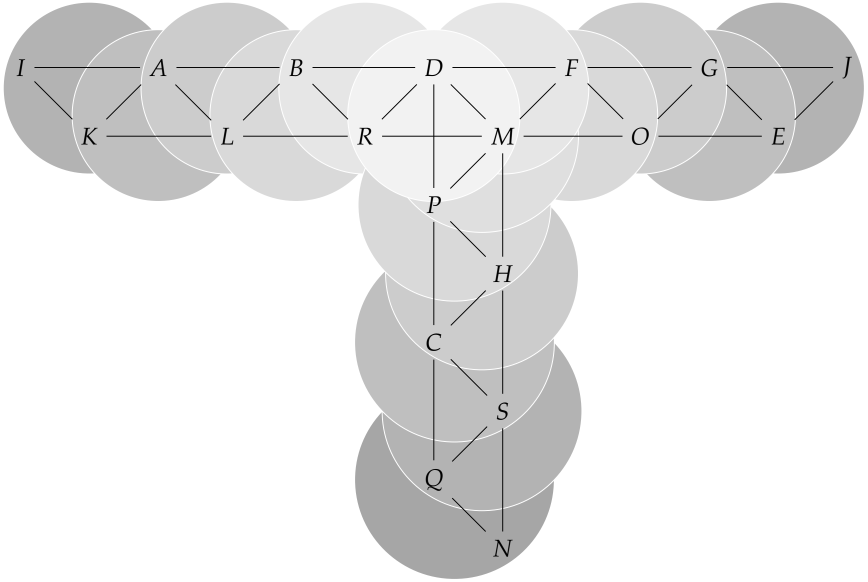
# Transition-based AMR parsing

- There has been previous work (Sagae and Tsujii; Damonte et al.; Zhou et al.; Ribeyre et al.; Wang et al.) on transition-based graph parsing.
- Our work introduces a new data structure "cache" for generating graphs of certain *treewidth*.

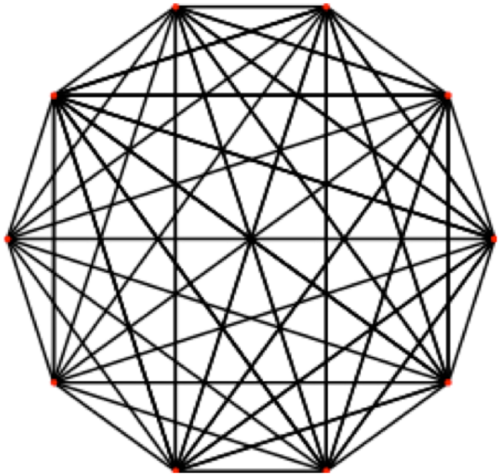
# Introduction to treewidth



A tree: treewidth 1

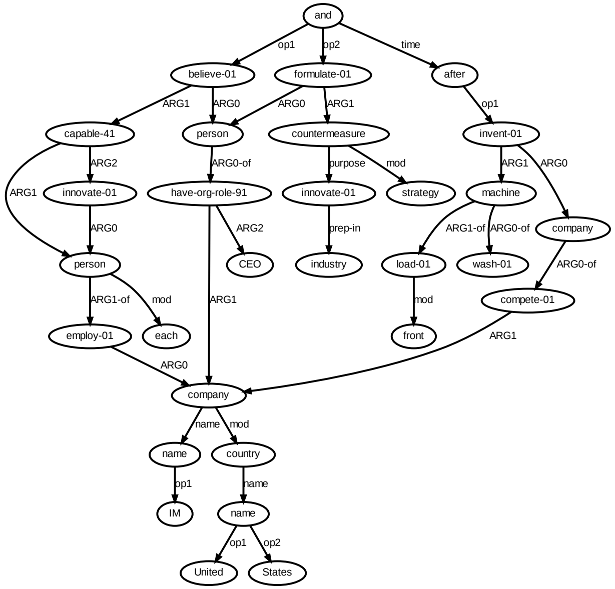


treewidth 2



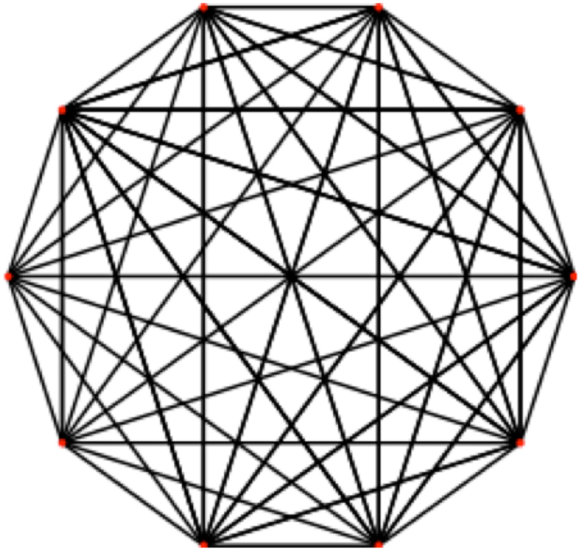
Complete graph of N nodes: treewidth N-1

# Introduction to treewidth



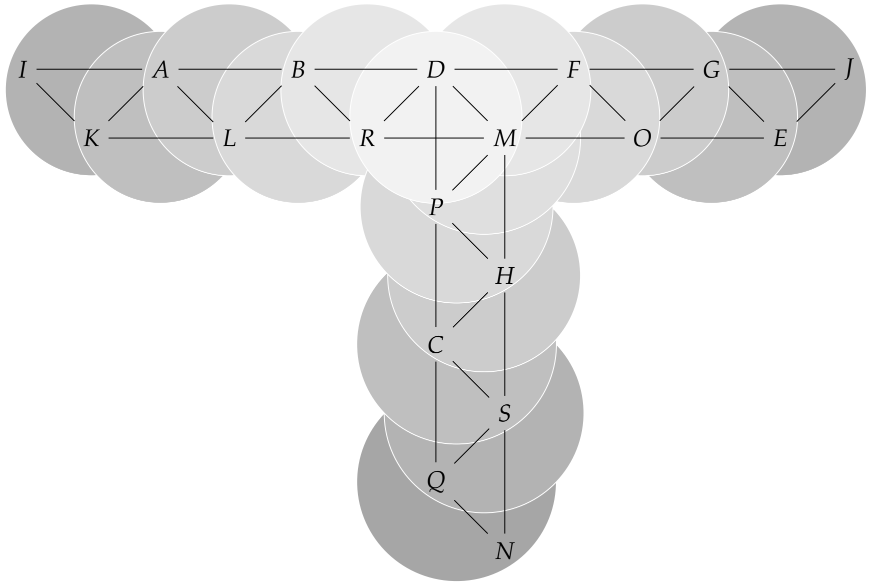
small tree width

~ 2.8 on average

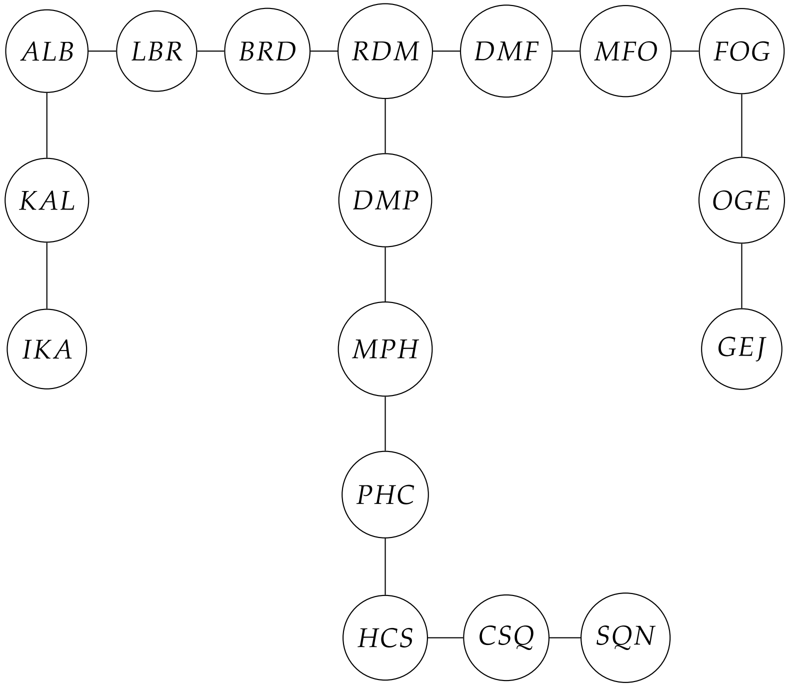


large tree width

# Tree decomposition



graph



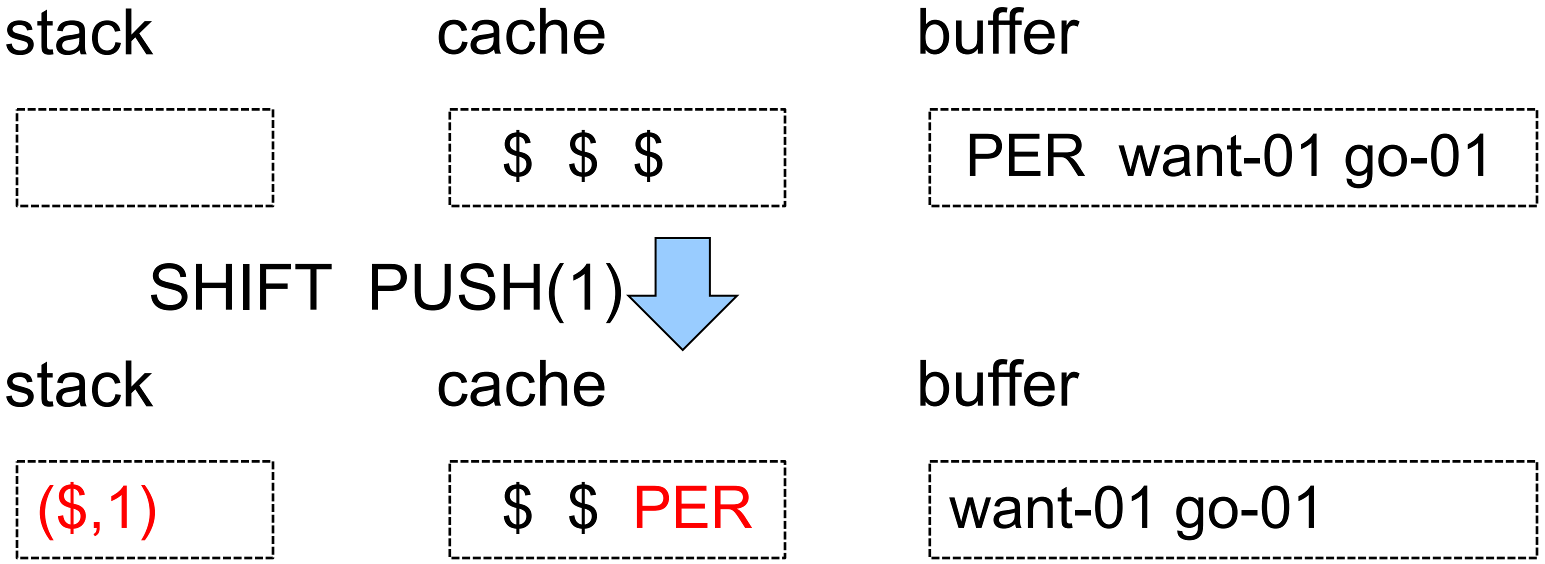
tree decomposition

# Cache transition system

- Configuration c $= (\sigma, \eta, \beta, E)$
  - Stack $\sigma$: place for temporarily storing concepts
  - **Cache η: working zone for making edges, fixed size corresponding to the treewidth.**
  - Buffer $\beta$: unprocessed concepts
  - E: set of already-built edges

# Cache transition system

- Actions
  - SHIFT PUSH(i): shift one concept from buffer to right-most position of cache, then select one concept (index i) from cache to stack.



stack

cache

buffer

$ $ $

PER want-01 go-01

SHIFT PUSH(1)

stack

cache

buffer

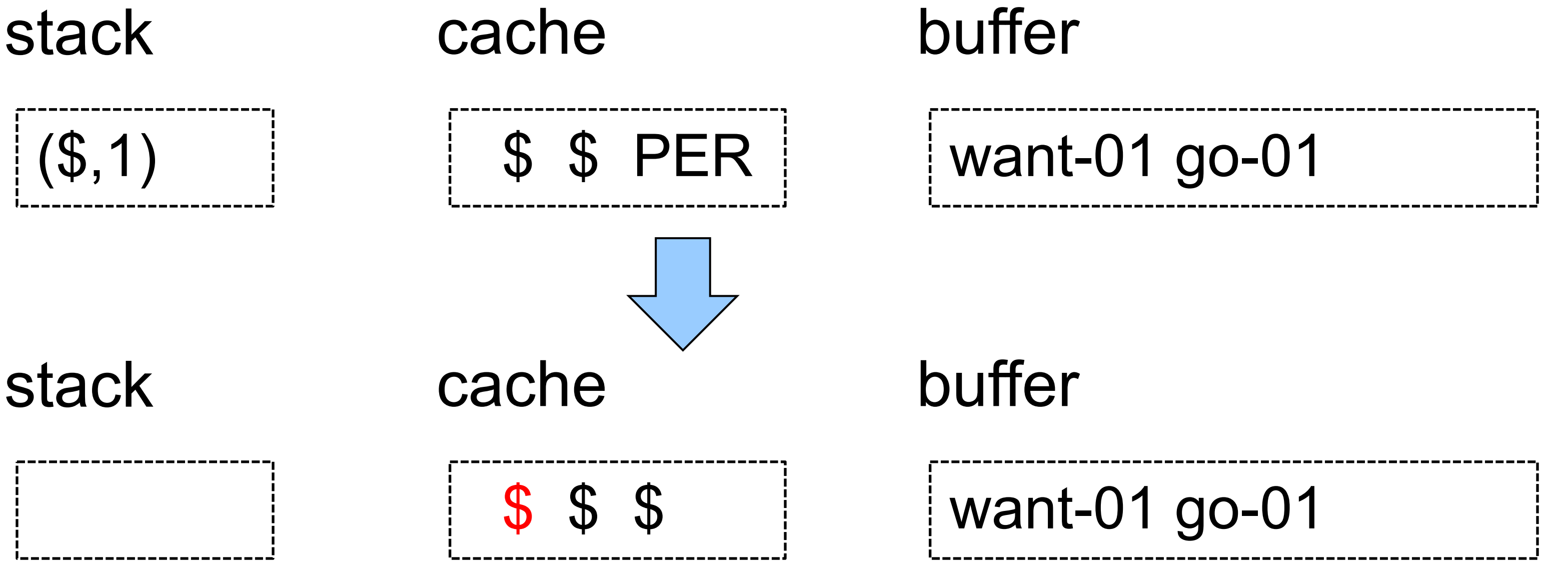
($,1)

$ $ PER

want-01 go-01

# Cache transition system

- Actions
    - POP: pop the top from stack and put back to cache, then drop the right-most item from cache.

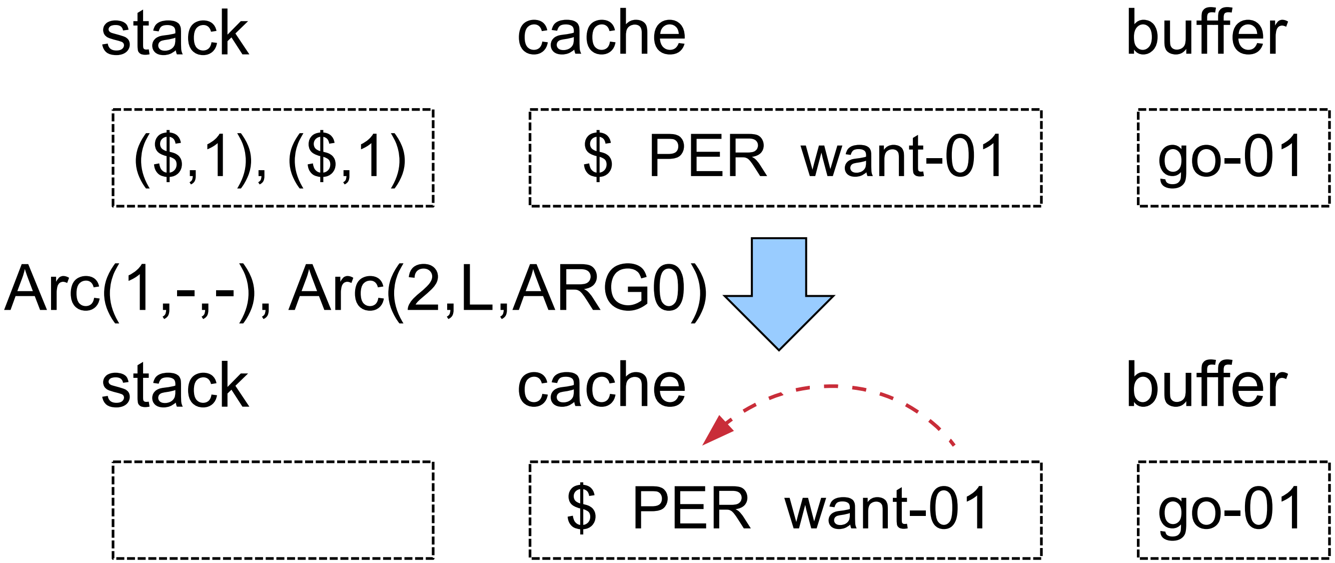


| stack | cache | buffer |
|---|---|---|
| ($,1) | $ $ PER | want-01 go-01 |

| stack | cache | buffer |
|---|---|---|
|  | $ $ $ | want-01 go-01 |

# Cache transition system

- Actions
  - Arc(i, l, d): make an arc (with direction d, label l) between the right-most node to node i. Arc(i,-,-) represents no edge between them.



stack

cache

buffer

($,1), ($,1)

$ PER want-01

go-01

Arc(1,-,-), Arc(2,L,ARG0)

stack

cache

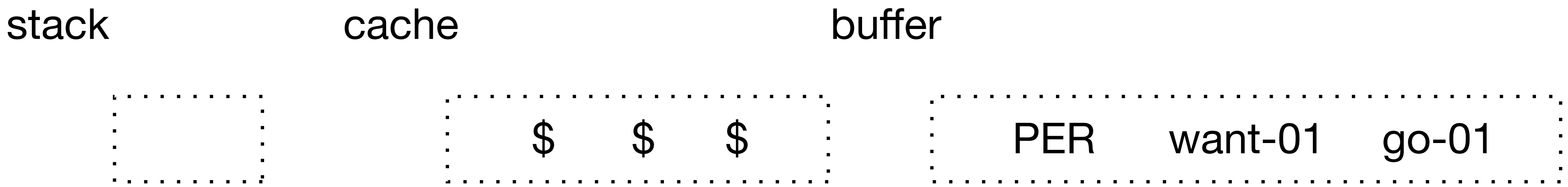
buffer

$ PER want-01

go-01

# Example of cache transition

Action taken: Initialization



stack

cache

$ $ $

buffer

PER want-01 go-01

# Example of cache transition

Action taken: SHIFT, PUSH(1)

stack

(1, $)

cache

$ $ PER

buffer

want-01 go-01

Hypothesis: PER

# Example of cache transition

Action taken: Arc(1, -, -), Arc(2, -, -)

| stack | cache | buffer |
|---|---|---|
| (1, $) | $ $ PER | want-01 go-01 |

Hypothesis: PER

# Example of cache transition

Action taken: SHIFT, PUSH(1)

stack

(1, $) (1, $)

cache

$ PER want-01

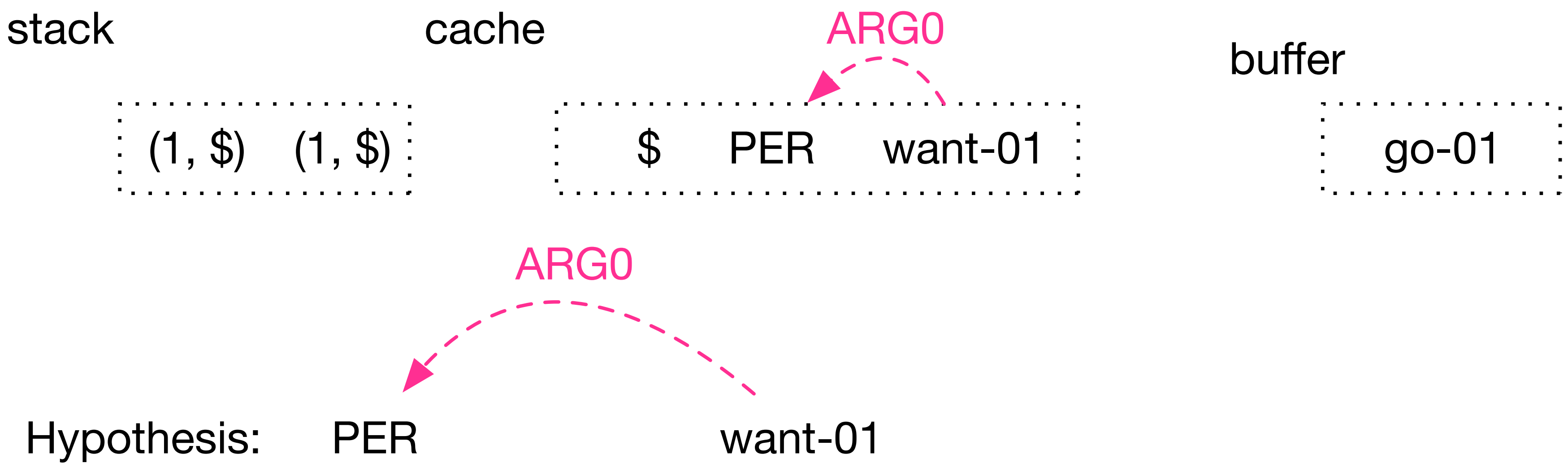
buffer

go-01

Hypothesis: PER want-01

# Example of cache transition

Action taken: Arc(1, -, -), Arc(2, L, ARG0)



stack

(1, $) (1, $)

cache

$ PER want-01

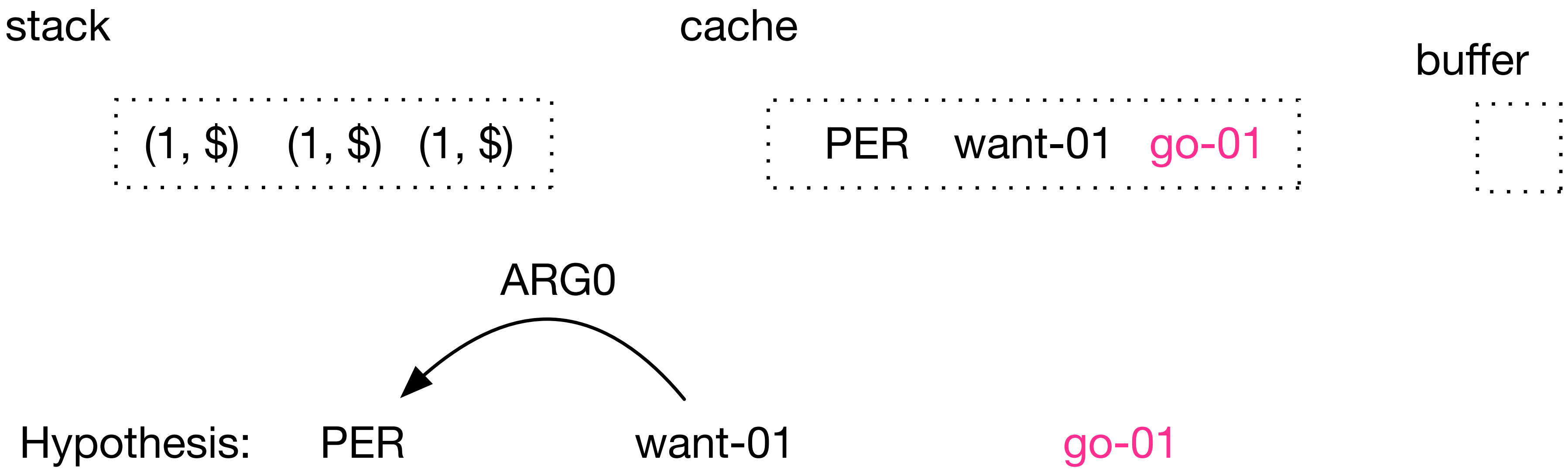
buffer

go-01

Hypothesis: PER want-01

# Example of cache transition

Action taken: SHIFT, PUSH(1)



stack

(1, $) (1, $) (1, $)

cache

PER want-01 go-01

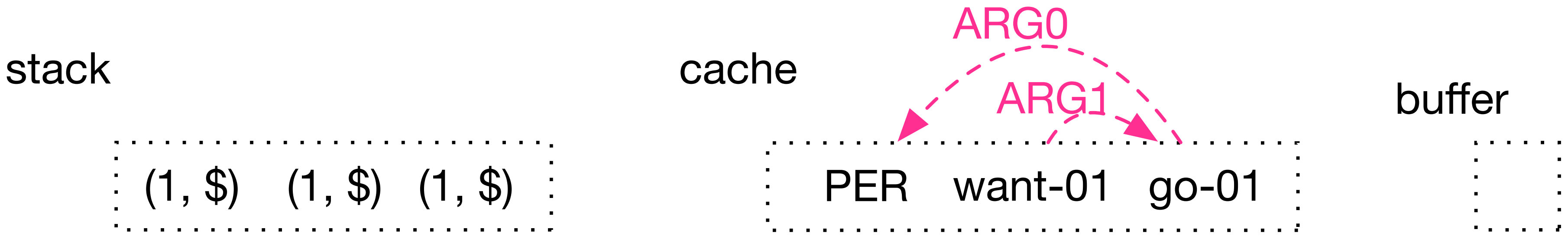
buffer

ARG0

Hypothesis: PER want-01 go-01

# Example of cache transition

Action taken: Arc(1, L, ARG0), Arc(2, R, ARG1)



stack

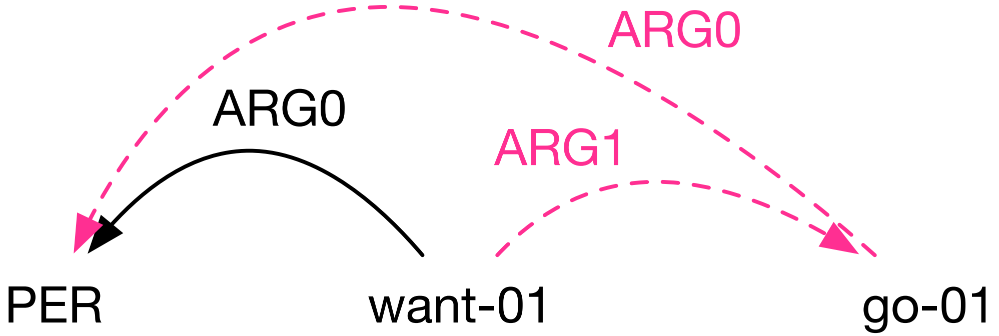
(1, $) (1, $) (1, $)

cache

PER want-01 go-01

buffer

Hypothesis:



# Example of cache transition

Action taken: POP POP POP

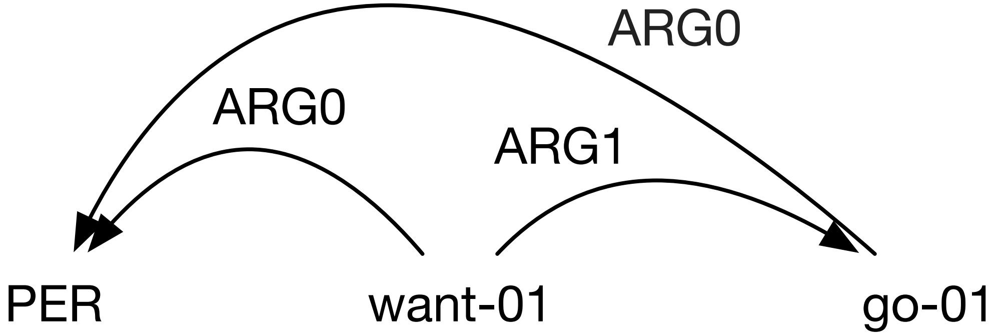
stack

cache

buffer

$ $ $

Hypothesis:



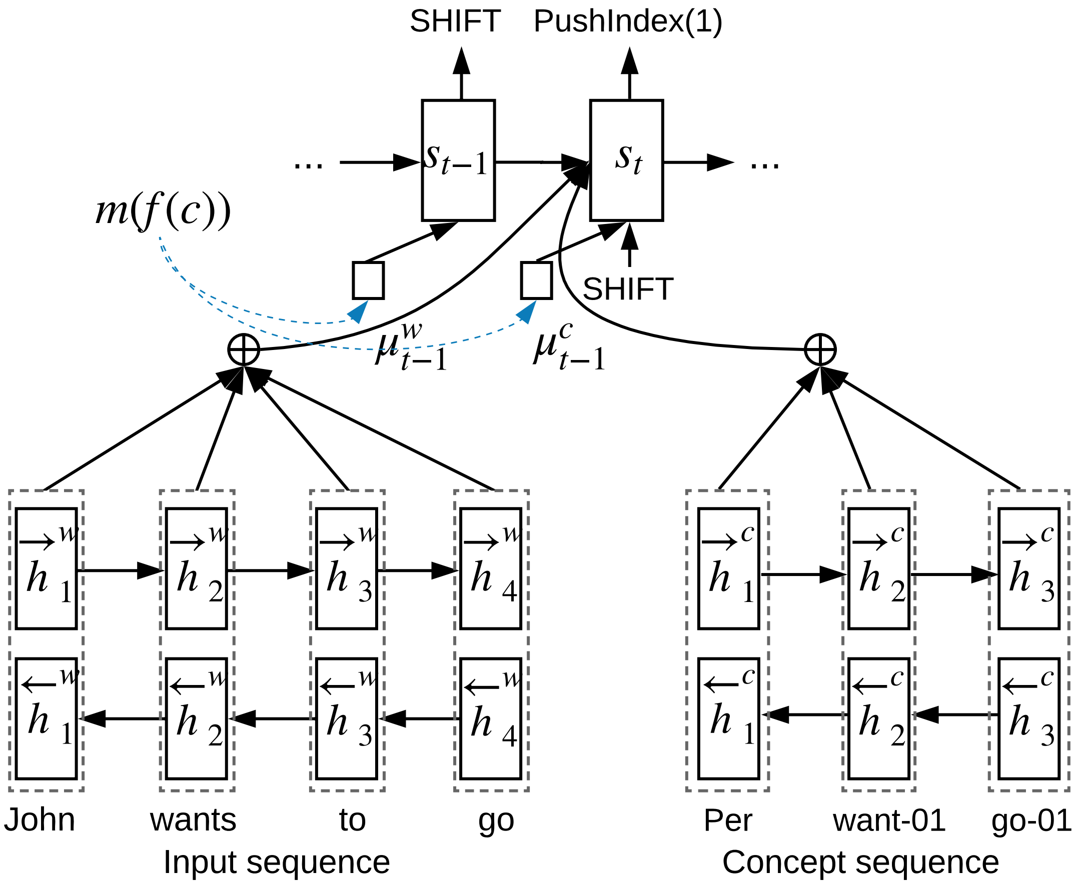
# Sequence to sequence models for cache transition system

- Concepts are generated from input sentences by another classifier in the preprocessing step.

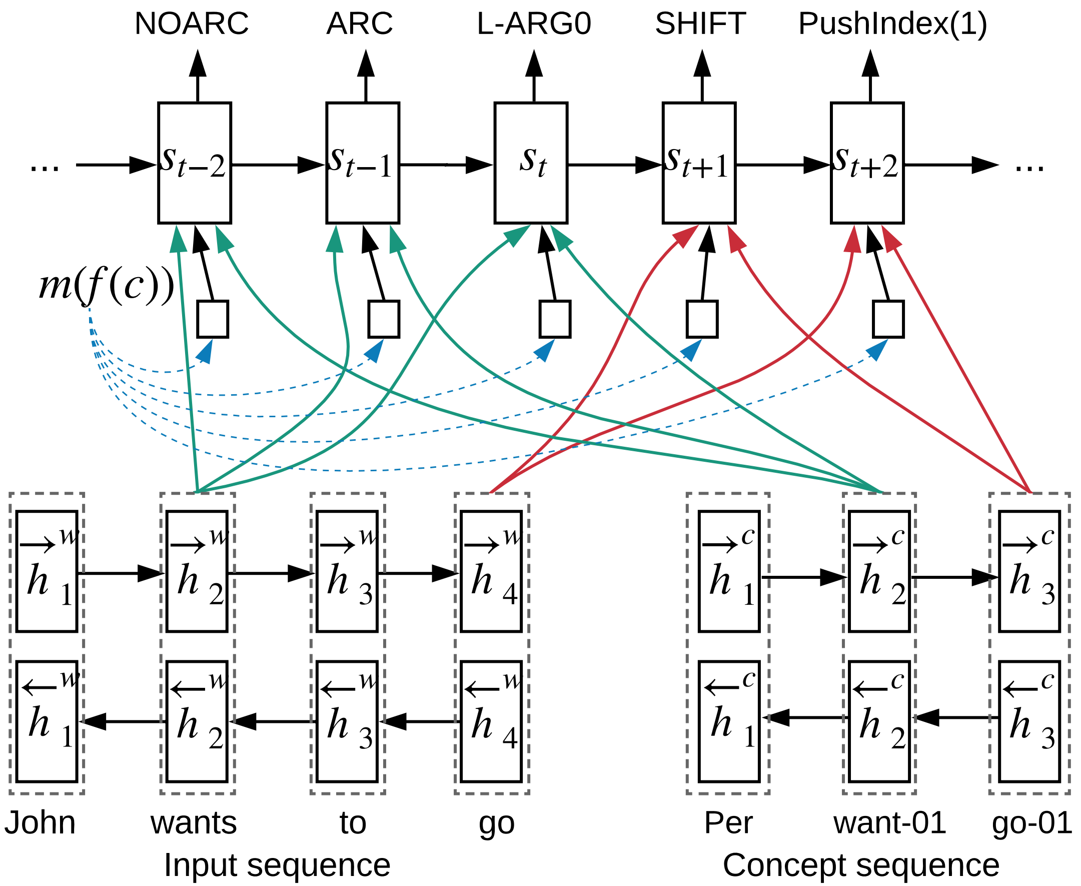
- Separate encoders are adopted for input sentences and sequences of concepts, respectively.

- One decoder for generating transition actions.

# Seq2seq (soft-attention+features)



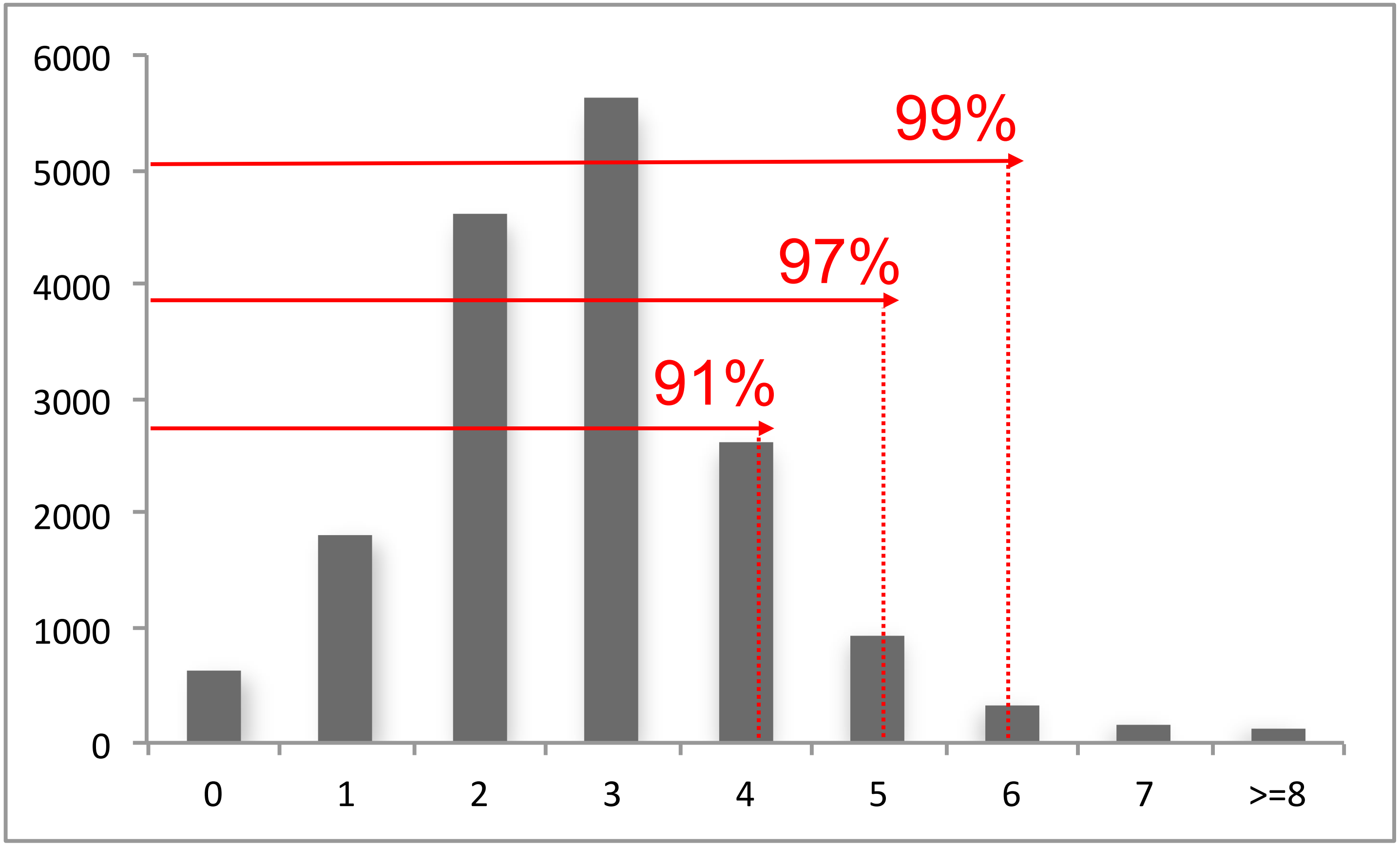
# Seq2seq (hard-attention+features)



# Experiments

- Dataset: LDC2015E86
    - 16,833(train)/1,368(dev)/1,371(test)
- Evaluation: Smatch (Cai et al., 2013)

# AMR Coverage with different cache sizes



# Development results

| Model | P | R | F |
|---|---|---|---|
| Soft | 0.55 | 0.51 | 0.53 |
| Soft+feats | 0.69 | 0.63 | 0.66 |
| Hard+feats | 0.70 | 0.64 | 0.67 |

Impact of various components

| cache size | P | R | F |
|---|---|---|---|
| 4 | 0.69 | 0.63 | 0.66 |
| 5 | 0.70 | 0.64 | 0.67 |
| 6 | 0.69 | 0.64 | 0.66 |

Impact of cache size

# Main results

| Model | P | R | F |
|---|---|---|---|
| Buys and Blunsom (2017) | -- | -- | 0.60 |
| Konstas et al. (2017) | 0.60 | 0.65 | 0.62 |
| Ballesteros and Al-Onaizan (2017) | -- | -- | 0.64 |
| Damonte et al. (2016) | -- | -- | 0.64 |
| Wang et al. (2015a) | 0.70 | 0.63 | 0.66 |
| Flanigan et al. (2016) | 0.70 | 0.65 | 0.67 |
| Wang and Xue (2017) | 0.72 | 0.65 | 0.68 |
| Lyu and Titov (2018) | -- | -- | 0.74 |
| Soft+feats | 0.68 | 0.63 | 0.65 |
| Hard+feats | 0.69 | 0.64 | 0.66 |

# Accuracy on reentrancies

| Model | P | R | F |
|---|---|---|---|
| Peng et al., (2018) | 0.44 | 0.28 | 0.34 |
| Damonte et al., (2017) | -- | -- | 0.41 |
| JAMR | 0.47 | **0.38** | 0.42 |
| Hard+feats (ours) | **0.58** | 0.34 | **0.43** |

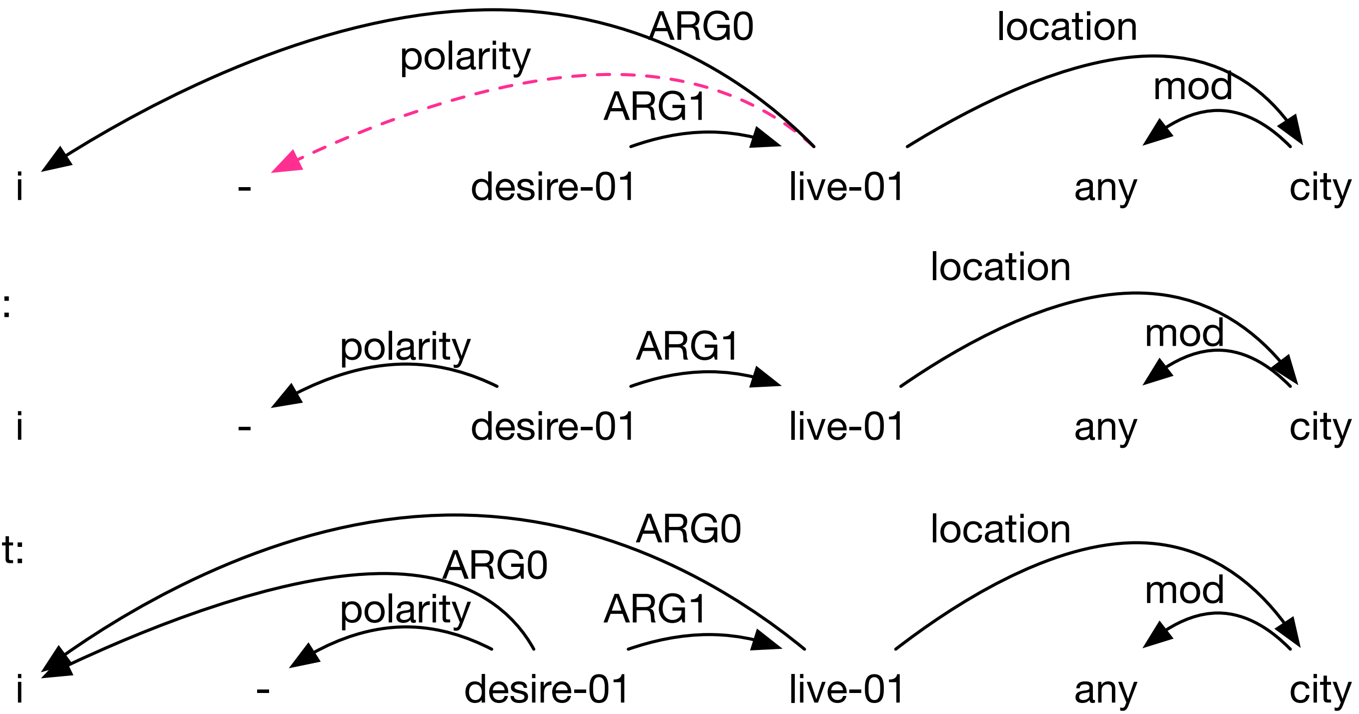
# Reentrancy example

Sentence: I have no desire to live in any city .

JAMR output:

Peng et al. (2018) output:

Our hard attention output:



# Conclusion

- Cache transition system based on a mathematical sound formalism for parsing to graphs.

- The cache transition process can be well-modeled by sequence-to-sequence models.
  - Features from transition states.
  - Monotonic hard attention.

# Thank you for listening!
# Questions