A Construction Grammar Parser in Soar

or

How I spent my sabbatical

John E. Laird

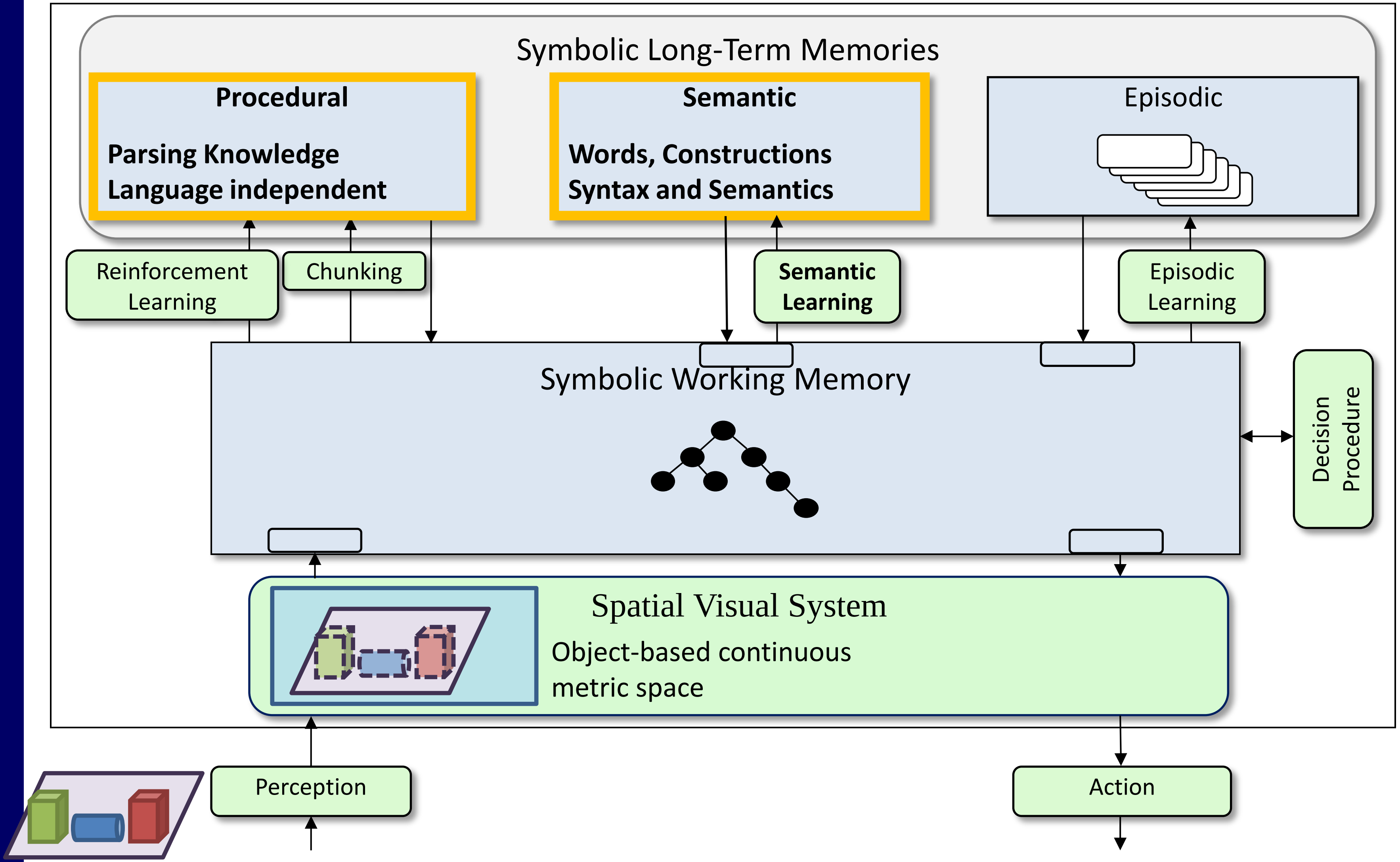
University of Michigan

June, 2014

Overview

- Goal of project
  - Flexible, extendable parser for interactive task learning
  - Use Construction Grammar
    - Theory of complex language usage and connections between syntax and semantics.
    - Syntax and semantics are associated with words, phrases, constructions
  - Use word by word, incremental repair-based parsing
    - Inspired by NL-Soar, XNL-Soar
    - Extend to constructions and word retrieval and ambiguity resolution
    - Integrate syntax, semantics, and pragmatic processing

Parsing Knowledge in Soar

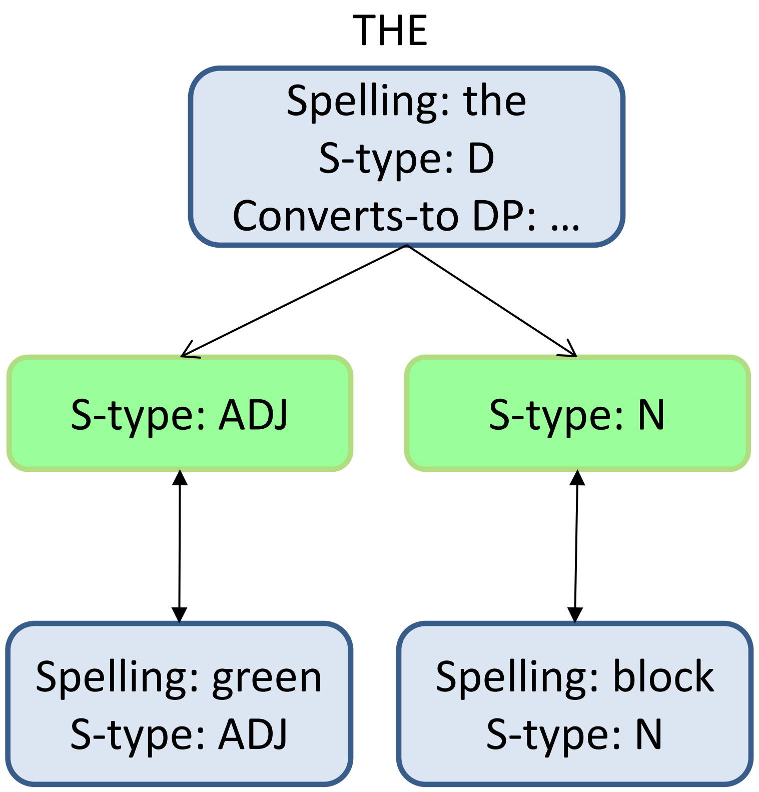


Example Sentences

1. Yes.
2. The block.
3. Open the pantry.
4. Store the green block.
5. This is a big triangle.
6. The big triangle is red.
7. The large one is red.
8. What is inside the pantry?
9. The red triangle is on the big green block.
10. Put the green sphere next to the pantry.
11. Pick up the green block on the stove.
12. Put this red triangle in the pantry.
13. Move the (medium) green block to the left of the large green block to the pantry.
14. Store the large green block on the red triangle.
15. Store the large green block (that is) on the red triangle.
16. Stack the red triangle, the medium block, and the large block.
17. All [of] [the] red triangles are in the pantry.
18. The chicken is cooked and the stove is off.
19. Wait until the chicken is cooked.

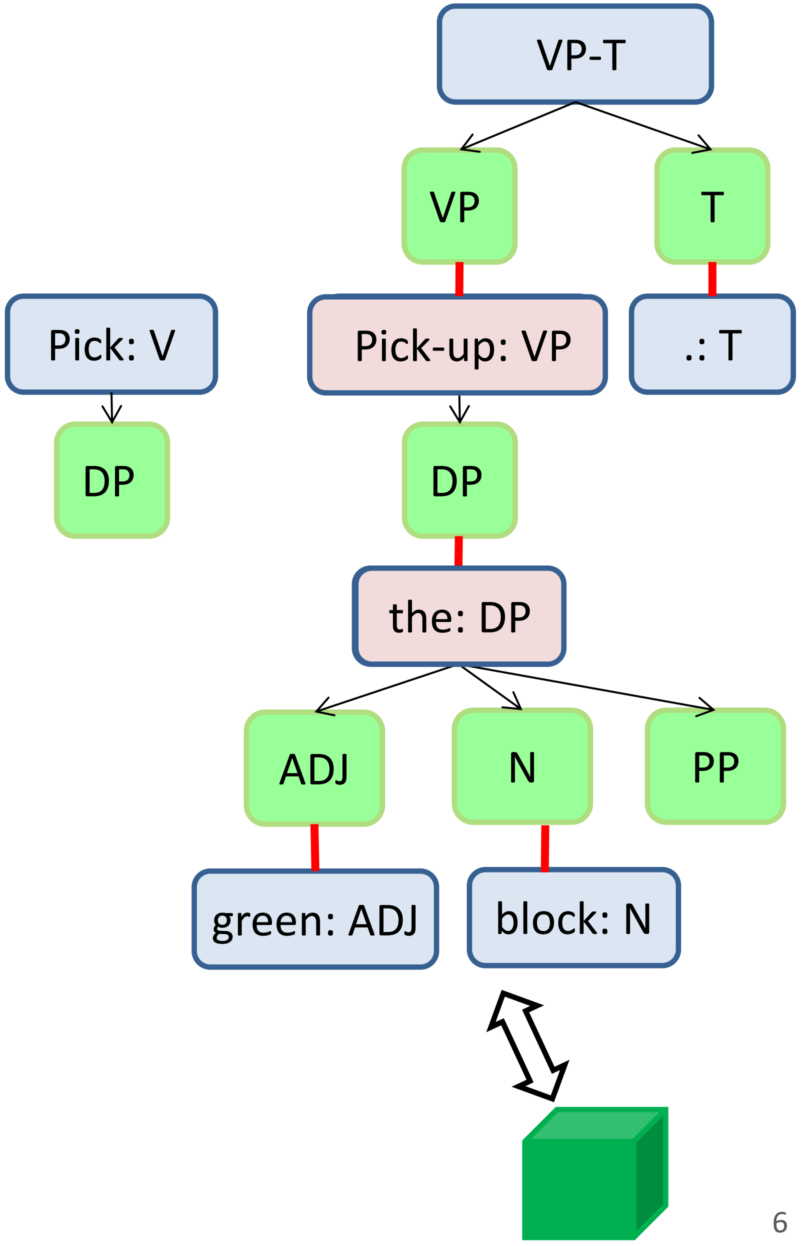
Basic Processing Idea
“The green block …”

1. Get a word
2. Find related structure from semantic memory and retrieve into working memory
   - Syntactic and semantic information about word
   - Possibly expectations (assigners) for other words/structures
   - Every word and structure has an S-type
     - Same as word sense for words
3. Merge assigners with retrieved structures
   - Builds up parse tree connecting structures.
4. Detect end of phrases
5. Build up associated semantics



Example Parse: Pick up the green block.

- comprehend-word (pick) [V]
- next-word (up)
- comprehend-word (up) [V->VP] [NP]
  - smem-retrieve (pick up)
  - remove-stale-structures (pick) [V]
- next-word (the)
- comprehend-word (the) [D->DP] [ADJ N]
- next-word (green)
- comprehend-word (green) [ADJ]
- merge [ADJ]
- next-word (block) [N]
- comprehend-word (block)
- merge [N]
- next-word (.)
- comprehend-word (.)
- process-phrase-end [D --> DP]
- ground-referent [green block: g01]
- merge [DP]
- process-semantics [object: g01]
- process-phrase-end [V --> VP]
- comprehend-construction [VP T]
- merge [VP]
- process-semantics (^action @p21)
- merge [T]
- terminate-processing



Simple words stored in Semantic Memory

- Most adjectives, adverbs, and nouns

```
(<block> ^spelling |block|
         ^structure-type N
         ^number singular
         ^object-feature shape
         ^perceptual-feature block1)
```

- Contains syntactic and semantic information

How to Represent Complex Syntactic Structures?

- Standard approach is associate syntax structure and semantics with individual words (lexical items).
  - Most “structure” is in the verb.
- Difficult to have contextualized structure and semantics.
- Constructions provide more complex structures for organizing semantics and syntax.

IS Construction

- **`DP-is-ADV-ADJ/DP/PP/U`**
- “The blue sphere is not in the pantry.”

```
(<x> ^structure-type CP
     ^current-word IS-V
     ^prior-word DP
     ^message-type object-description
     ^assigners <DP> <IS> <ADV> <ADJ> <DP> <PP> <U>)

(<DP> ^structure-type DP
      ^syntactic-structure head
      ^semantic-structure object
      ^required true)

(<IS-V> ^structure-type IS-V
        ^syntactic-structure predicate
        ^semantic-structure assignment
        ^required true
        ^referent.id soar-assignment)

(<ADV> ^structure-type ADV ...)

(<ADJ> ^structure-type ADJ
       ^exclusive <DP> <PP> <U> ...)
```

Different “is” Constructions

- DP-is-ADJ/DP/PP/U
  - “The blue sphere is in the pantry.”
- ADJ-is-DP:
  - “Green is a color.”
- N-is-DP:
  - “Sphere is a shape.”
- This-is-DP/ADJ/PP:
  - “This is in the pantry.”
- What-is-PP-?:
  - “What is in the pantry?”
- Where-is-DP-?:
  - “Where is the red block?”

Challenge: Handling Ambiguity

- Word sense ambiguity:
  - Is “block” a noun or verb?
- Construction ambiguity
  - Which construction should be used for these words?
- Phrase attachment
  - Where should a new structure attach?

Word Sense Ambiguity Error Detection and Recovery

- The stove is on
  - `DP-is-ADV-ADJ/DP/PP/U`
  - If retrieve “on” as a adjective, that works great…
  - But “on” is really a preposition (but it doesn’t know)
- What to do?
  1. Find the most recent word that has type (word sense) that could merge with an existing assigner.
     - Every word sense also lists other possible word sense for the same spelling. Is this reasonable? Not sure.
  2. Retry the retrieval again, but block the first sense and add the ^type to the retrieval cue.
     - (<cue> ^spelling on ^type P)
  3. Remove old sense and add in new sense.

Retrieving Constructions

1. Constructions are schemas of types and words
   - Need context to retrieve correct construction
   - Want to bias to most specific
2. Hypothesis: use two elements for cued retrieval
   From specific to general
   - Word-Word: “pick” “up” (pick up --> pick)
   - Word: “is” -> IS-V; “are” -> IS-V; “was” -> IS-V, …
   - Type-Word: DP “is”
   - Word-Type: “Where” IS-V;
   - Type-Type: DP-ISV-ADJ/PP (The block is red)

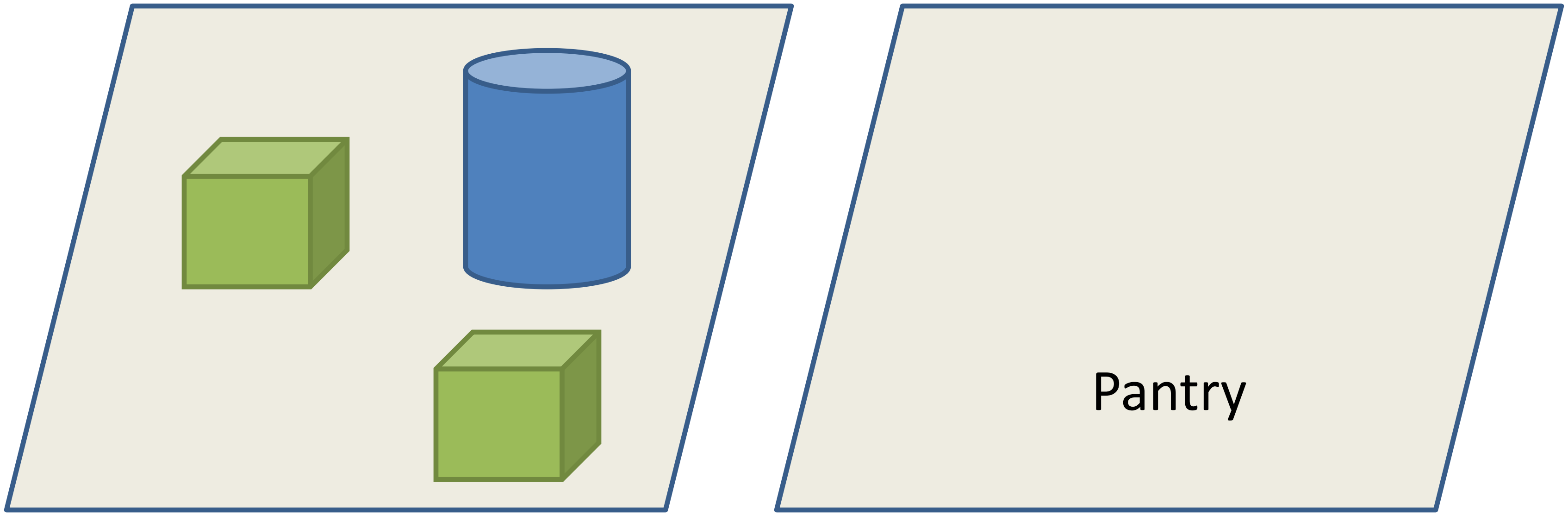
- Maybe spreading activation would do this more naturally…

Phrase Attachment

- Store the green block left of the cylinder.

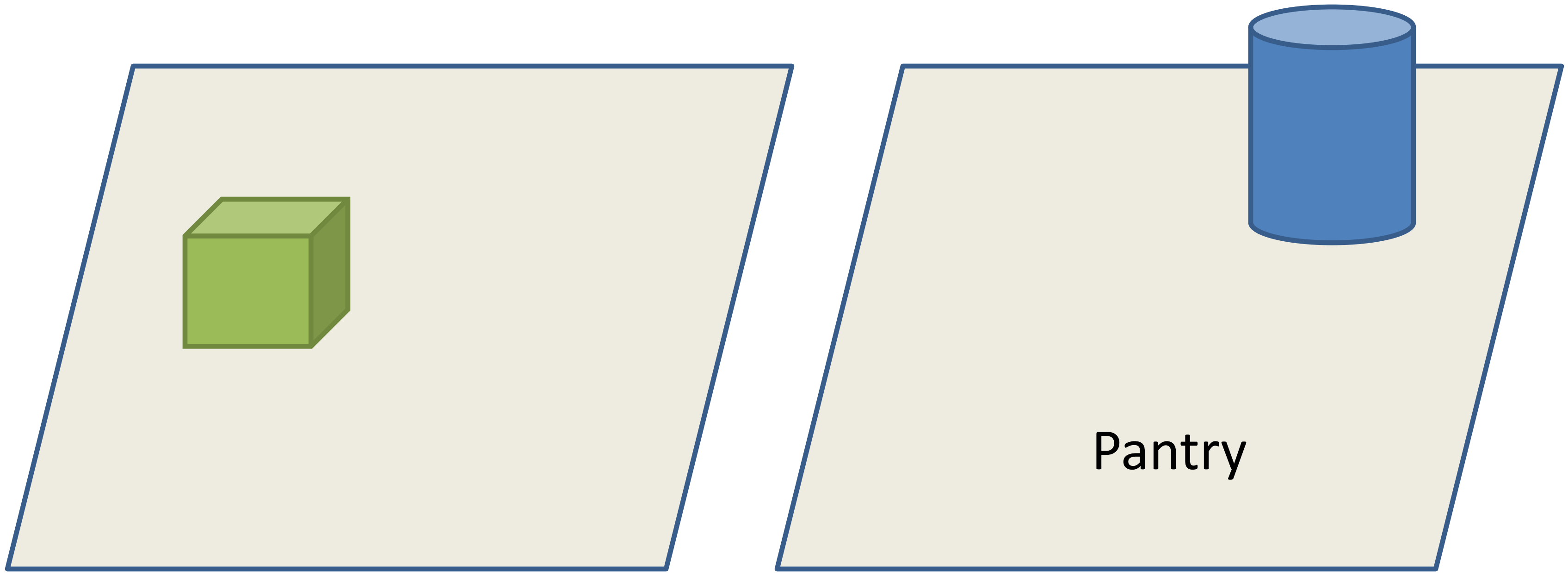
Phrase Attachment

- Store the green block [that is] left of the cylinder.



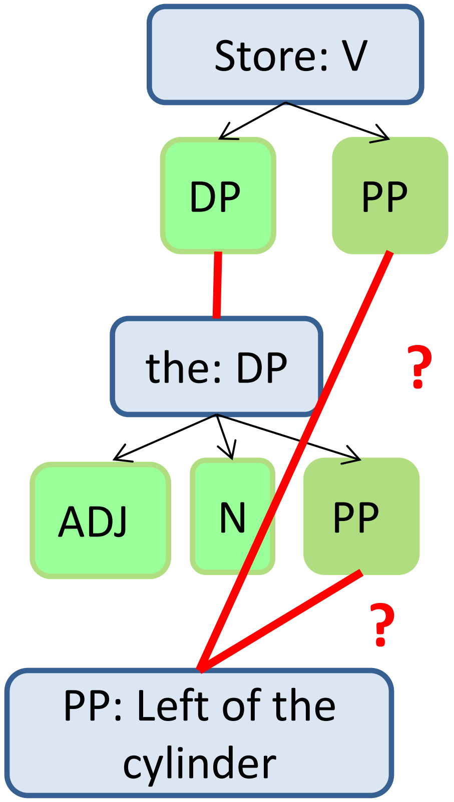
Phrase Attachment

- Store the green block [to the] left of the cylinder.



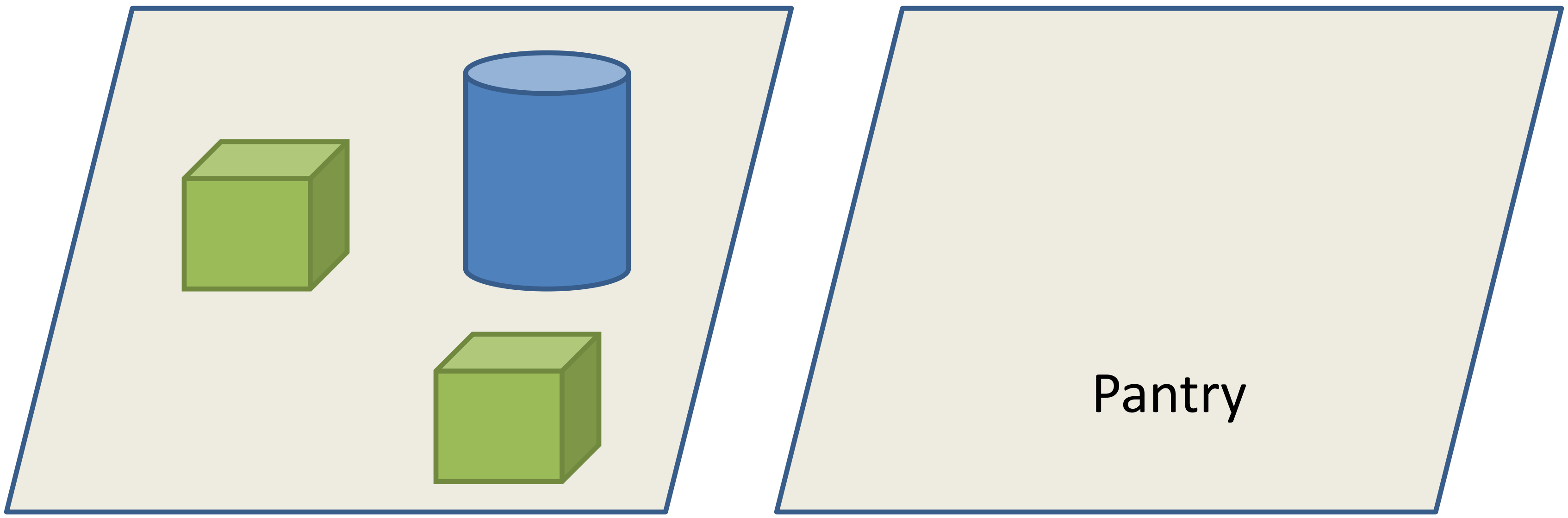
Phrase Attachment

- Store the green block left of the cylinder.
- Tie for two merge operators!
- Resolve it through look-ahead
  - Copy relevant structures to substate
  - Apply each merge in evaluate-operator
  - Attempt to ground relevant objects with new constraint
  - How do # of groundings change?
    - One with greatest decrease but > 0 wins
    - =information gain.
    - Usually only one succeeds…
    - If tie, use other heuristics…



Phrase Attachment

- Store the green block [that is] left of the cylinder.
  - Groundings go from 1 to 1.
- Store the green block [to the] left of the cylinder.
  - Groundings go from 1 to 0.



Some Data

- Parsing 33 sentences

- **Soar 9.3.2**
- Kernel CPU Time: 0.671 sec.
- Total CPU Time: 0.811 sec.
- 5478 decisions (0.122 msec/decision)
- 14979 elaboration cycles (2.734 ec's per dc, 0.045 msec/ec)
- 17224 inner elaboration cycles
- 4676 p-elaboration cycles (0.854 pe's per dc, 0.143 msec/pe)
- 26689 production firings (1.782 pf's per ec, 0.025 msec/pf)
- 136336 wme changes (68287 additions, 68049 removals)
- WM size: 238 current, 488.928 mean, 903 maximum

- **Soar 9.3.3**
- Kernel CPU Time: 0.555 sec.
- Total CPU Time: 0.591 sec.
- 5513 decisions (0.101 msec/decision)
- 15064 elaboration cycles (2.732 ec's per dc, 0.037 msec/ec)
- 17336 inner elaboration cycles
- 4689 p-elaboration cycles (0.851 pe's per dc, 0.118 msec/pe)
- 26776 production firings (1.777 pf's per ec, 0.021 msec/pf)
- 137016 wme changes (68627 additions, 68389 removals)
- WM size: 238 current, 490.244 mean, 903 maximum

Nuggets & Coal

- Nuggets
  - Left to right parser of construction grammar
  - Uses multiple words for construction/meaning retrieval
  - Multiple retrievals to repair if “local” error detected for word/construction retrieval
  - Propositional attachment done naturally using combination of preferences and one-step look-ahead
  - Fast…
- Coal
  - Not completely integrated with Rosie!
  - Many additional types of constructions to test!
  - Only does simple examples of word meaning re-retrieval!
  - Doesn’t use chunking!
  - No evaluation!
  - Lots of arbitrary design decisions.