Impact of Working Memory Activation on Agent Design

John Laird,

University of Michigan

What’s the Problem?

- How do you write a Soar agent so it can survive automatic removal from working memory activation decay?
- Many problems arise from the violation of Soar’s close-world assumption
  - Soar assumes that if something isn’t in working memory, it is not true, whereas it could be it just haven’t retrieved it from semantic memory.

Working Memory Activation & Semantic Memory Reminders

1. Only applies to WME's with long-term identifiers.
   - Don't have to worry about random structures disappearing.
2. Complete object is removed from working memory
   - Not individual WME's [which violated closed-world assumption even more!]
3. Always use *mirroring* with semantic memory
   - Keeps WM in sync with SMEM without deliberate saves
   - If something is removed from WM it is in SMEM

Mirroring WME's

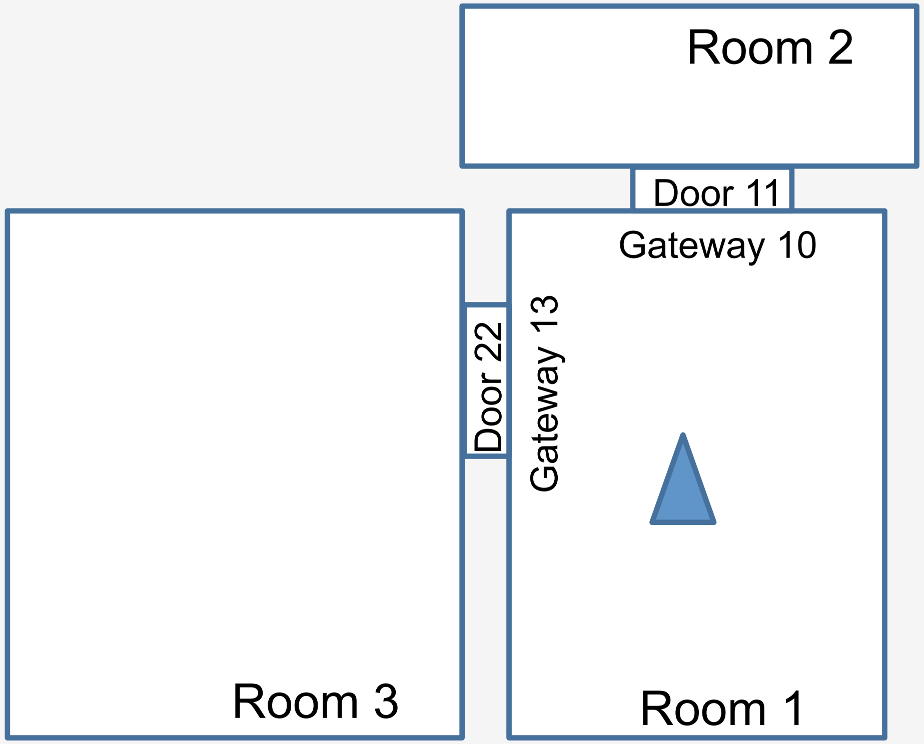
When something a WME is added to a LTI, then that WME is added to semantic memory.

- Eliminates need for most deliberate saves.

For a graph structure (such as a topological map)

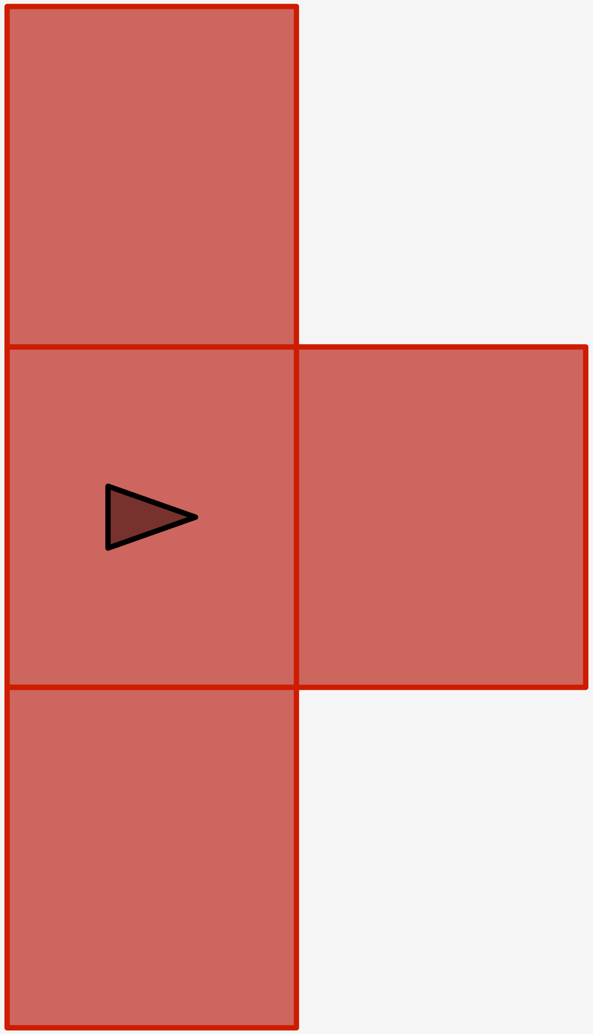
1. Deliberately save the root to SMEM.
2. All incremental substructure is automatically added to SMEM.

Example From Robot World

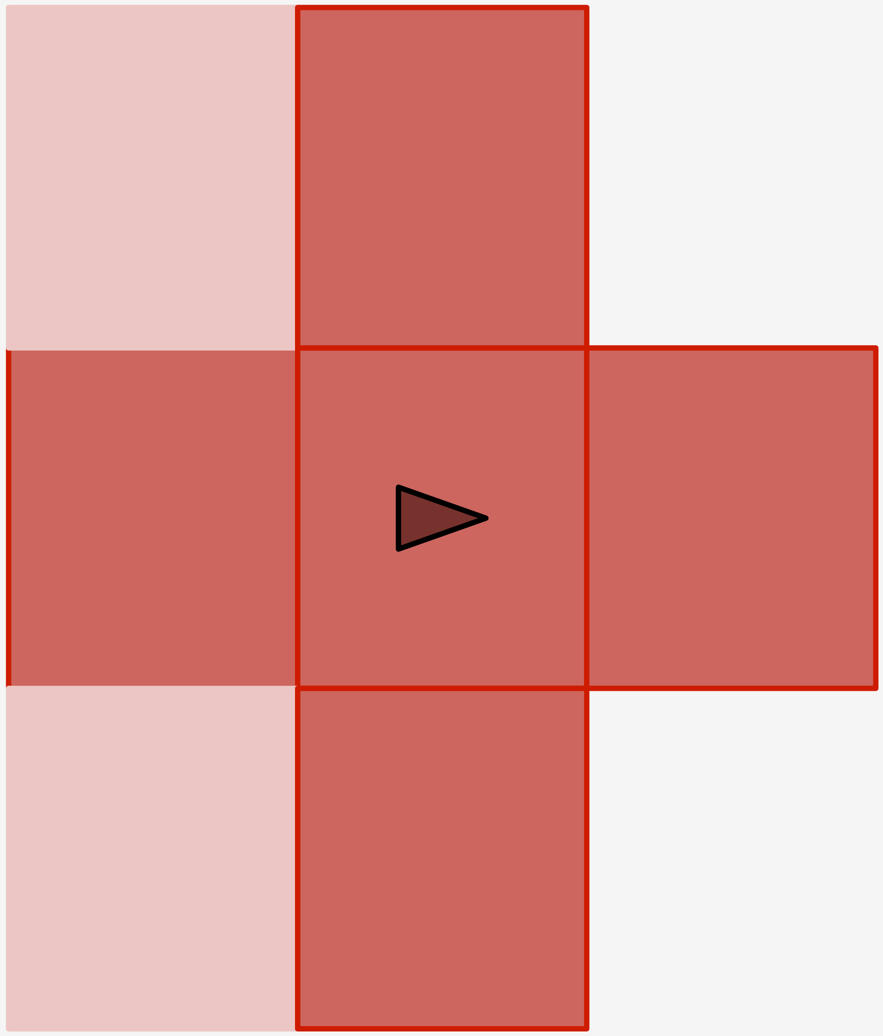


```
(state ^current-location @A3)
(@A3 ^id 1 ^type room
    ^wall @W1 @W2 @W3 @W4
    ^gateway @G1 @G2
    ^neighbor @N1 @N2
    ^next @R1 @R2)
(@W1 ^id 33 ^type wall ^x 34 ^y 44
    ^dir south)
…
(@G1 ^id 10 ^type gateway
    ^x 35 ^y 64 ^to 11)
…
(@N1 ^id 11 ^type doorway
    ^wall @W5 …
    ^gateway …
    …)
…
(@R1 ^id 2 ^type room
    …)
```

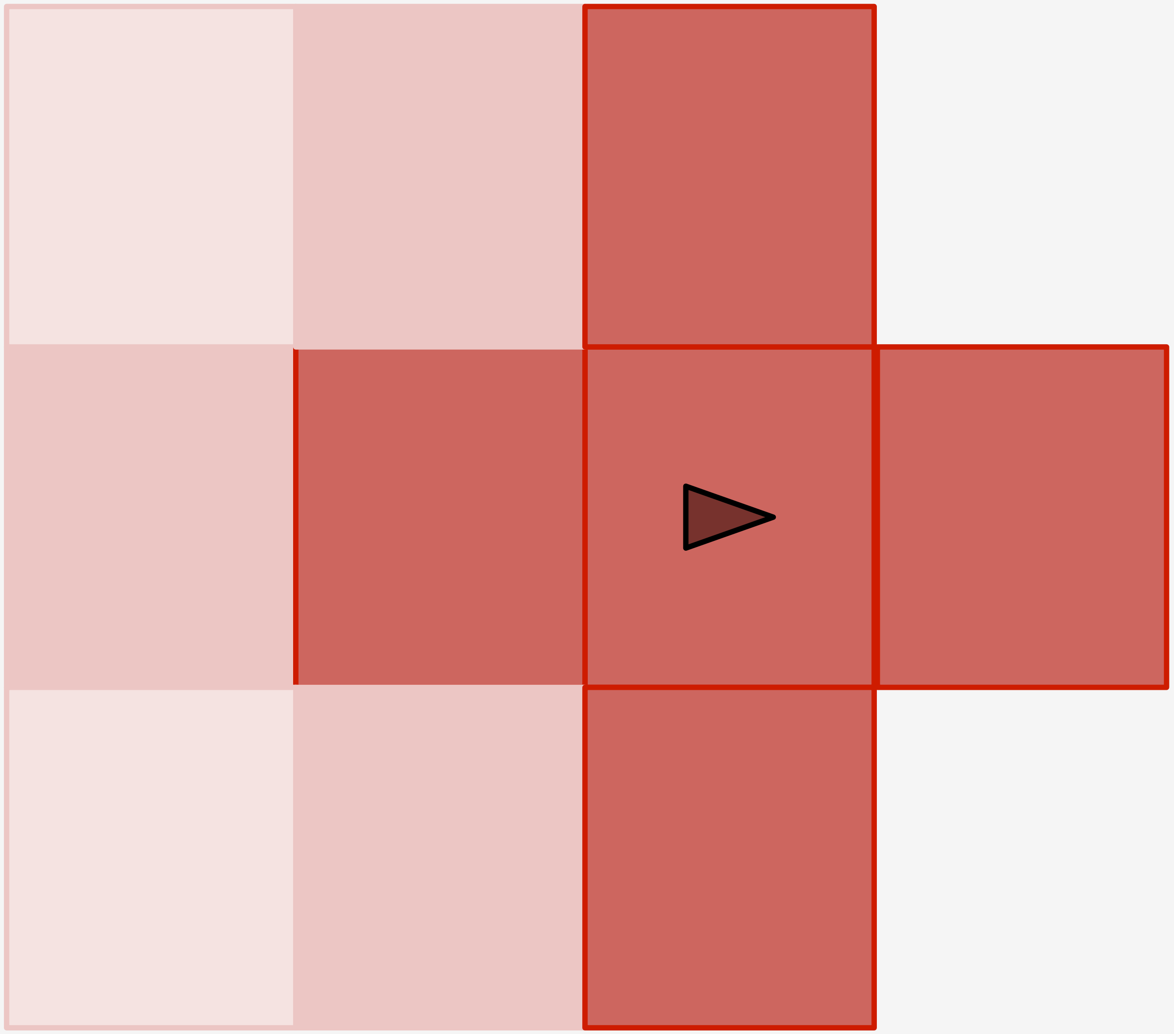
Desired Behavior



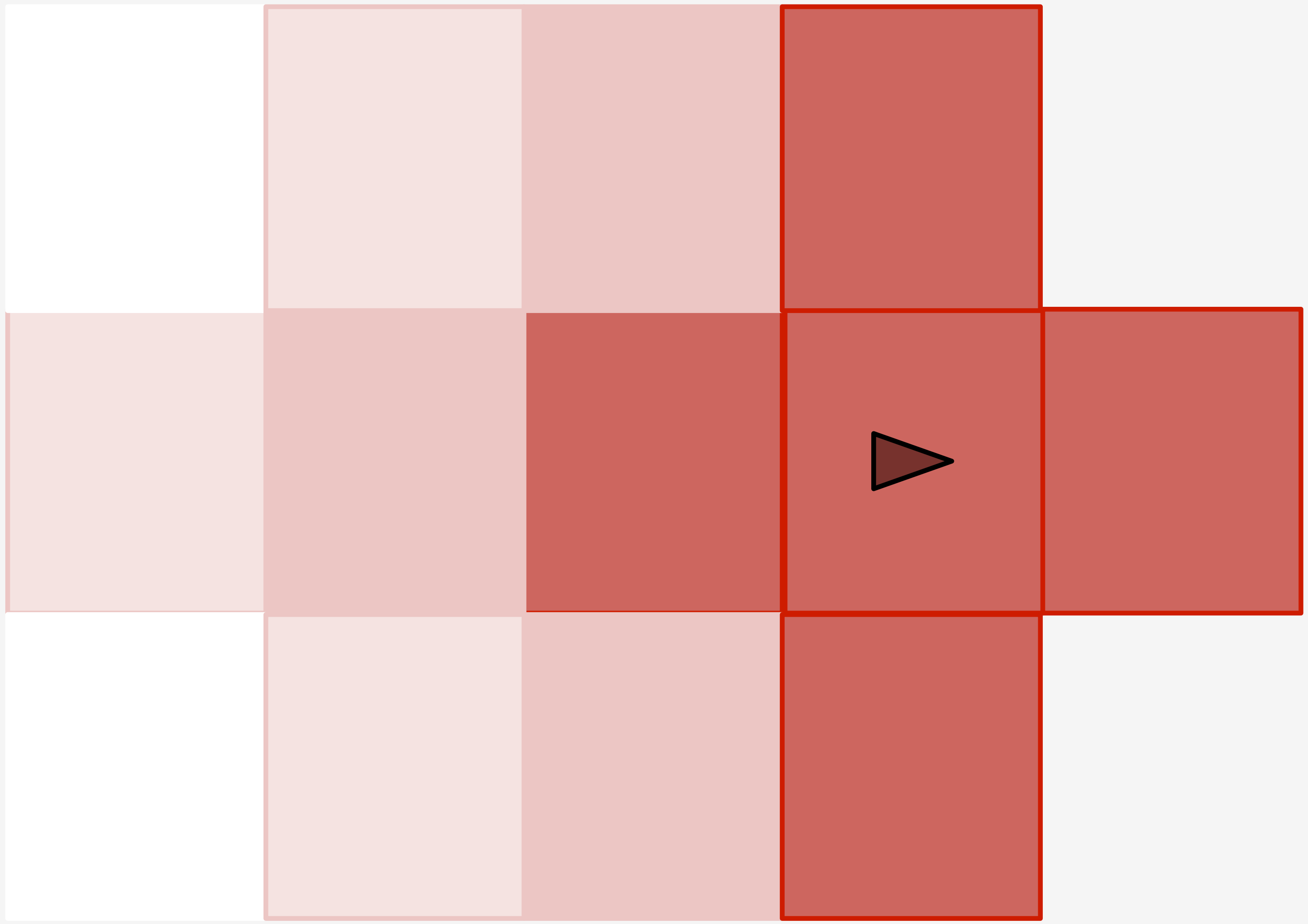
Desired Behavior



Desired Behavior



Desired Behavior

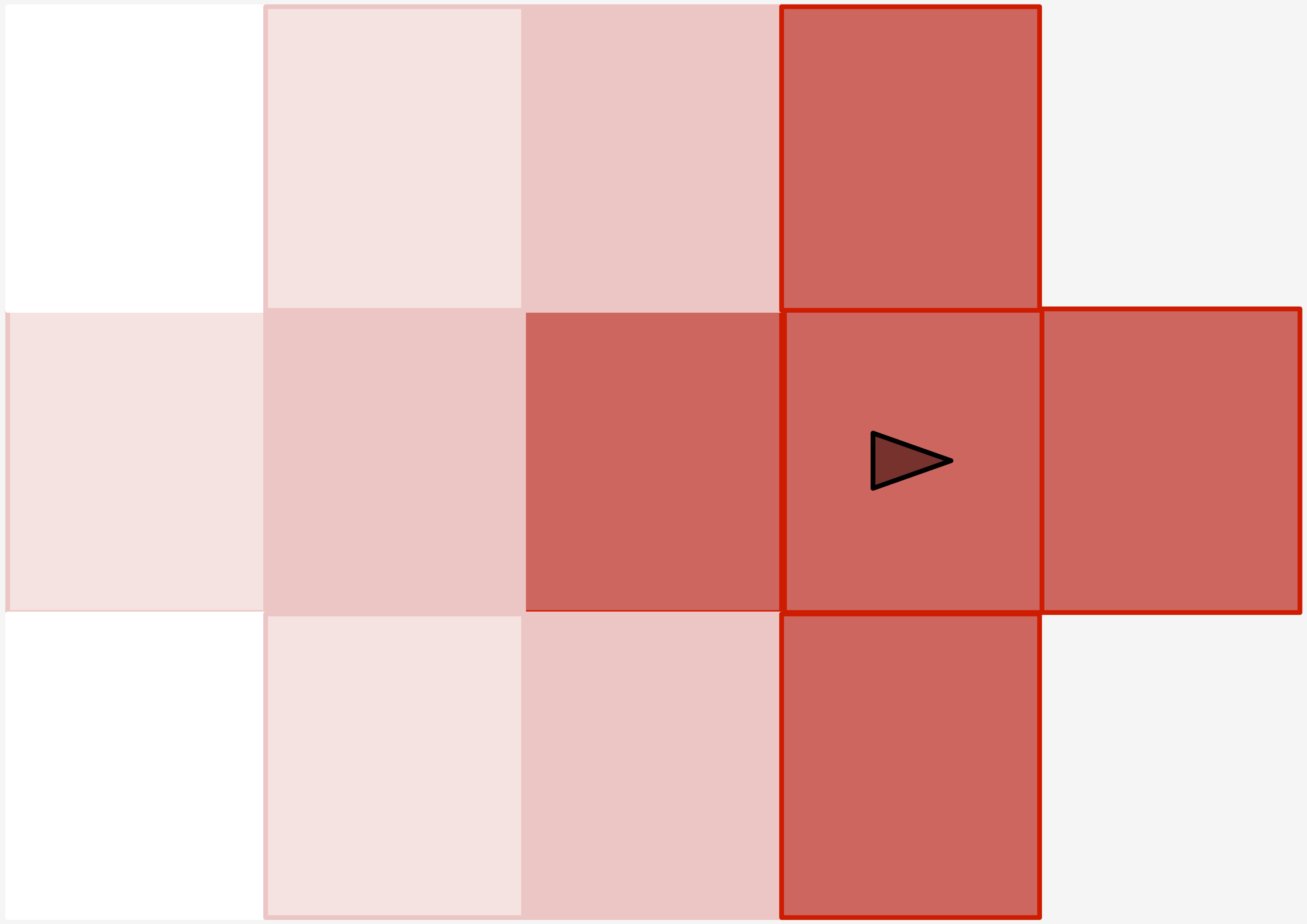


Problem #1 - Retrieval

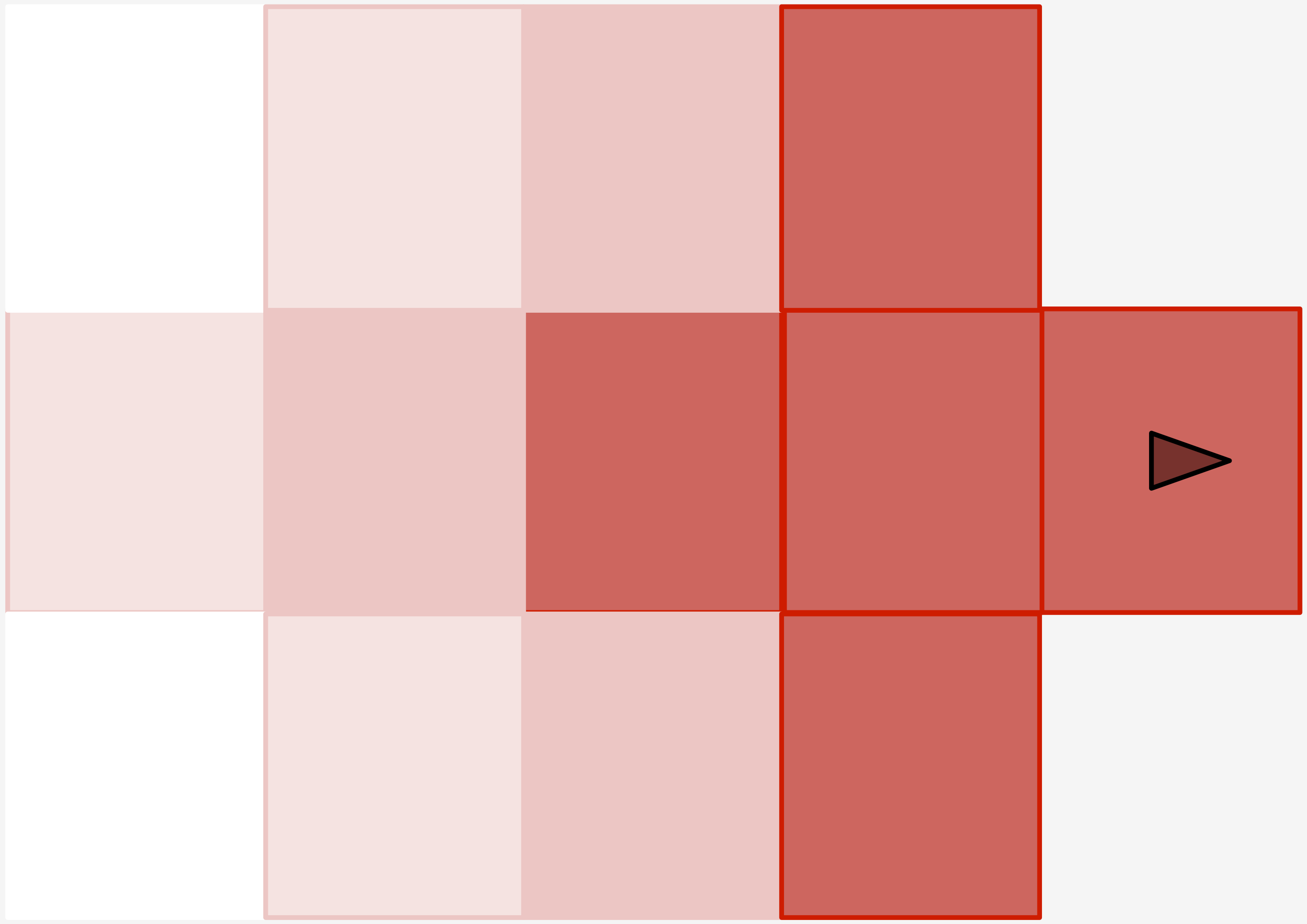
How does an agent maintain what it needs in working memory?

1. Proactive Retrieval
   – Maintain “cloud” of possibly relevant structures.
2. Reactive Retrieval
   – Retrieve structures when don’t know what to do.

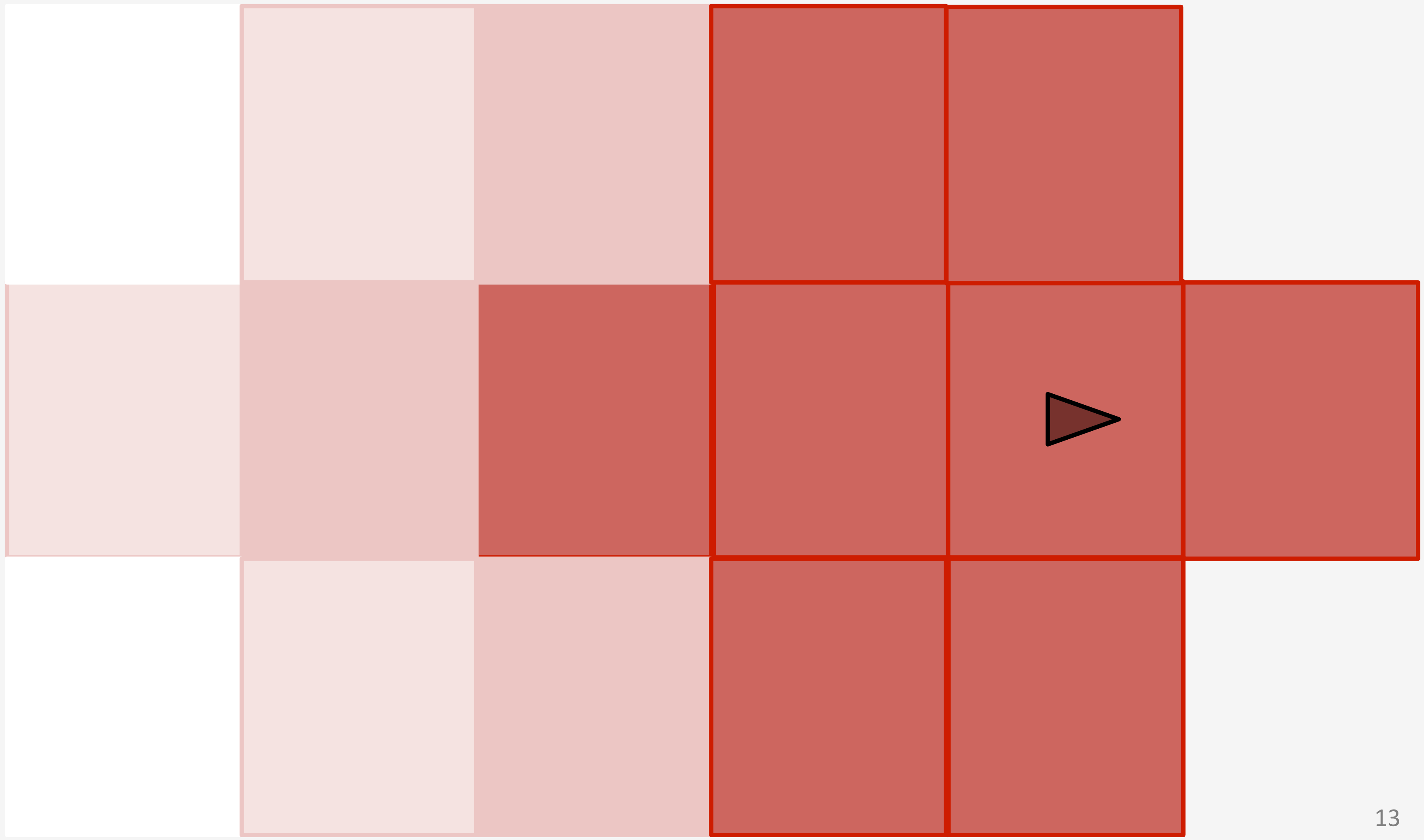
Proactive



Proactive



Proactive



Proactive Retrieval

1. Manually identify those structures and substructures that are necessary for making decisions in current state.
   - `If trying to reach the storage room, and the storage room is next to the current room, go to the gateway that leads to the storage room.`
2. If ever those structures are missing, select an operator to retrieve them.
   - `If the next room is missing, then retrieve it.`

```
sp {xxx
     (state <s> ^name substate-name
                ^current-location.next <next>)
     (<next> -^id)
     -->
     (<s> ^operator <o> +, =, >)
     (<o> ^lti-retrieve <next>)}
```

   - If don't prefer operator, agent might select a task operator based on incomplete information.

Representation Lesson

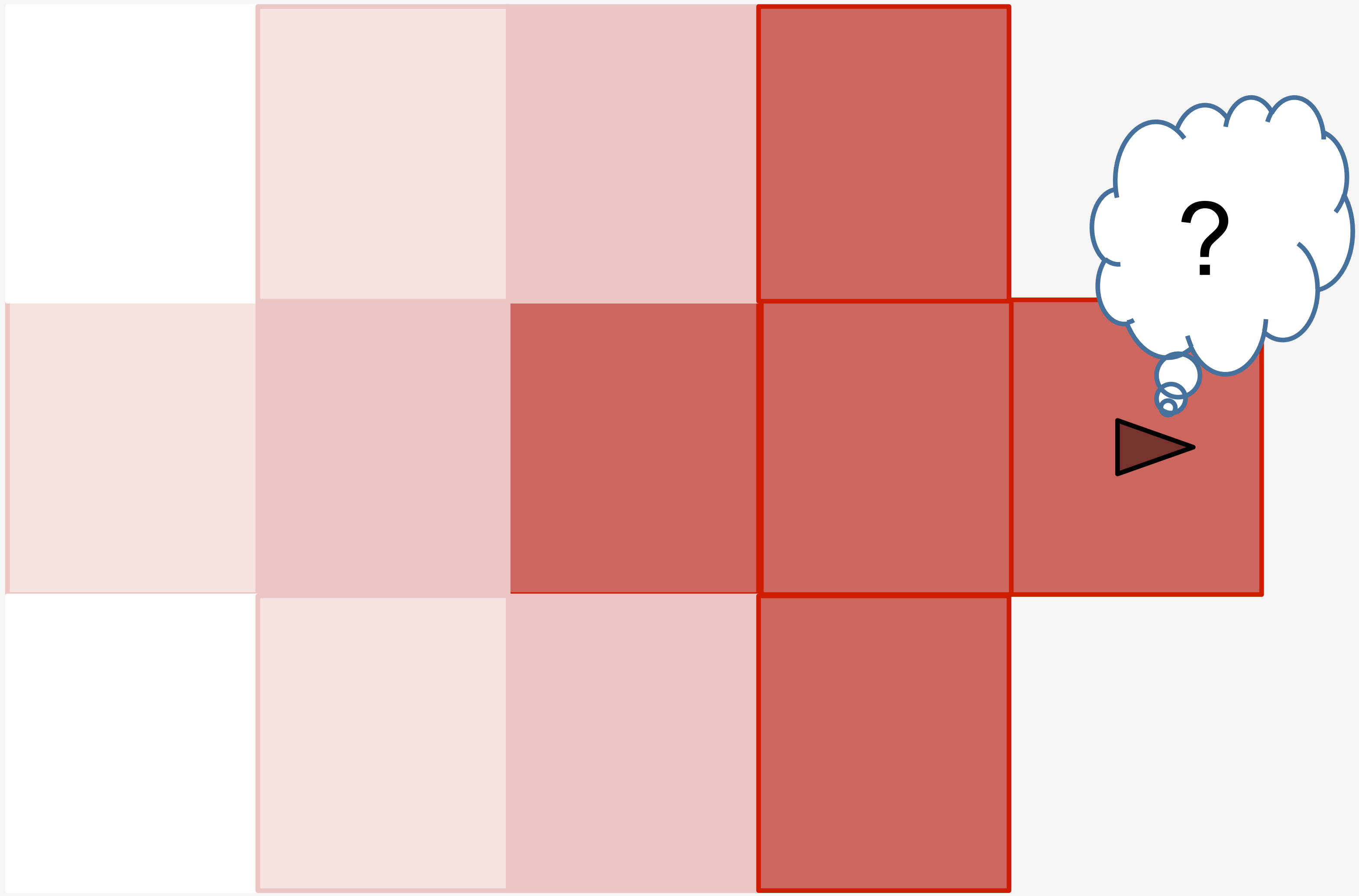
- Must have common attribute (`^id`) with unique values for objects so can check if structure is in working memory.
- Otherwise will keep trying to retrieve even if successful.

Reactive Retrieval

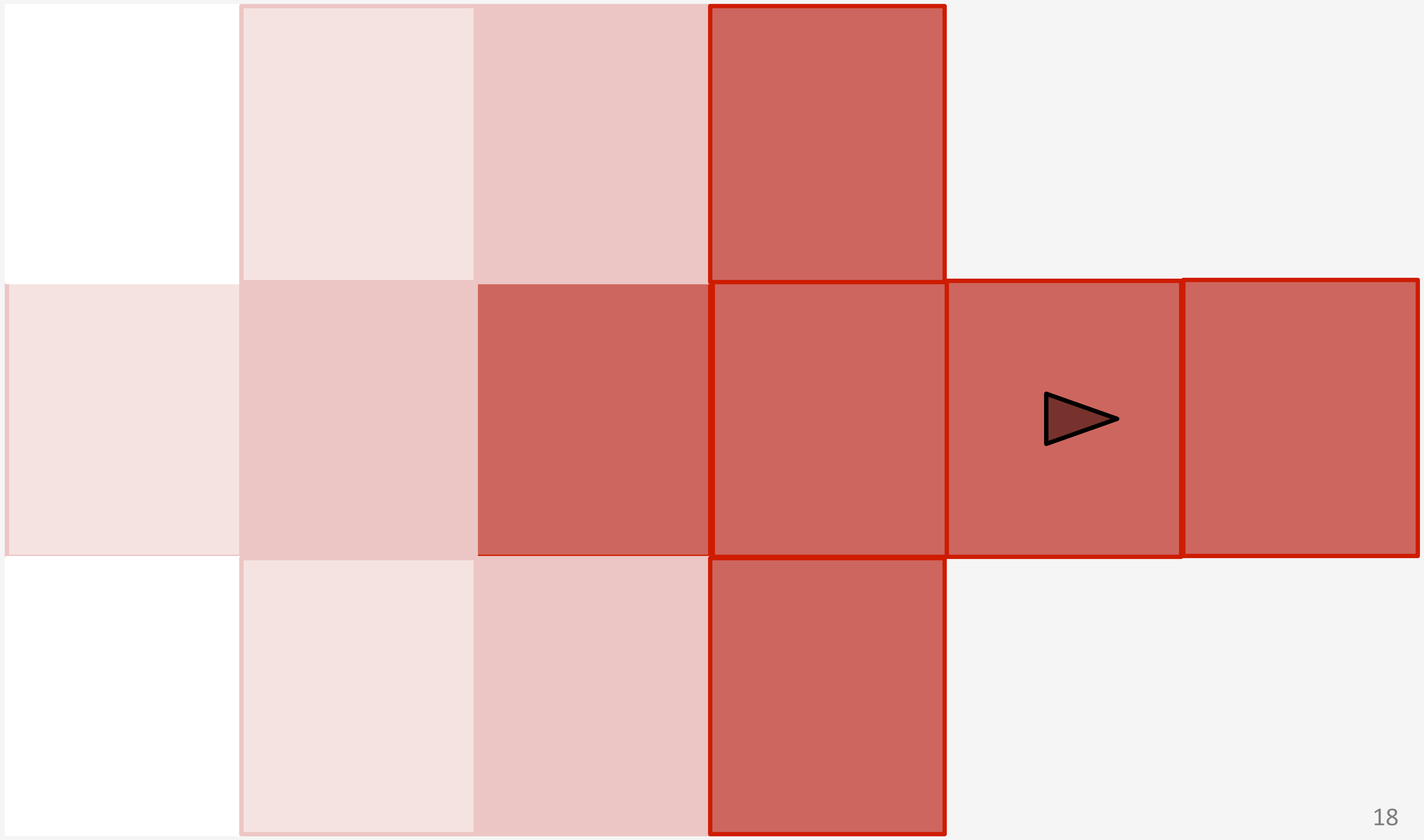
1. If agent can't make progress without information, retrieve when hits state no-change.
   - If trying to move to a gateway, and don't have information on where it is, will impasse when try to select operators to move to it.

```
sp {xxx
    (state <s> ^impasse no-change
               ^attribute state
               ^superstate <ss>)
    (<ss> ^name go-to-gateway
          ^destination-gateway <gw>)
    (<gw> -^id)
   -->
    (<s> ^operator <o> +, =)
    (<o> ^lti-retrieve <gw>)}
```

Proactive



Reactive



Discussion

- Reactive is more elegant
  - Only done when needs it
  - Can imagine task-independent rules that retrieve missing structures
  - But relies on complete failure to make progress.
- I don't believe it is possible to design agent so that only it needs reactive retrieval, but beyond scope of this talk.

Problem #2: Incremental Creation (avoid duplicate creation)

1. When creating a representation of a new room in working memory (and mirrored in SMEM), must use operators to create structures.
   - Need operators because room structure (wall, gateways) must be persistent and because there might be variable numbers of substructure.
2. Ambiguity when a wall structure has been removed.
   - Don't know if a missing wall is in SMEM (can't test id).
   - Should a new wall be created or should retrieve wall?
3. Solution
   - Record on room structure the id of a substructure when it is created: ^wall-id 34 ^gateway-id 45, …
   - Only create new wall if its id is not already created.

Rules for Recording Walls

```
sp {robot*propose*record-smem-new-wall
   (<s> ^name robot
        ^current-location <cloc>
        ^io.input-link.area <a>)
   (<cloc> ^id <aid>
          -^wall-id <wid>)
   (<a> ^id <aid>
        ^wall <wall>)
   (<wall>  ^id <wid>)
-->
   (<s> ^operator <op> + =)
   (<op> ^name record-smem-new-wall
         ^wall <wall>)}
```

```
sp {apply*record-smem-current-area-wall
   (state <s> ^operator <op>
              ^current-location <cloc>)
   (<cloc> ^id <aid>)
   (<op> ^name record-smem-new-wall
         ^wall <wall>)
   (<wall> ^x <x>
           ^y <y>
           ^id <id>
           ^direction <dir>)
-->
   (<cloc> ^wall <nw1>
           ^wall-id <id>)
   (<nw1>  ^type wall
           ^x <x>
           ^y <y>
           ^id <id>
           ^direction <dir>)}
```

Problem #3

- If takes too long to perform internal subtask, may continual forget and never complete
  1. During extended internal planning, agent can forget original structures
  2. Recalling structures might blow away intermediate results
- Approaches:
  1. Do retrievals in substates – sometimes difficult
  2. Cheat – add rules whose purpose is to keep structures active (redundant elaboration rules).
  3. Use chunking (eliminates need for lots of intermediate structures)
  4. Use different search methods (progressive deepening!!!).

Problem #4: Negation (closed world)

- Negations don't help activation!
  - If the only tests of an attribute are negations, it will decay and be removed.
  - Structures that *prevent* action do not receive activation.

- Can't have a lone negation of an object:
  - `(<x> -^attribute value)`
    - Can't tell if does not have that value, or object has decayed
  - Always test id as well as rest of negation.
  - `(<x> ^id <id> -^attribute value)`

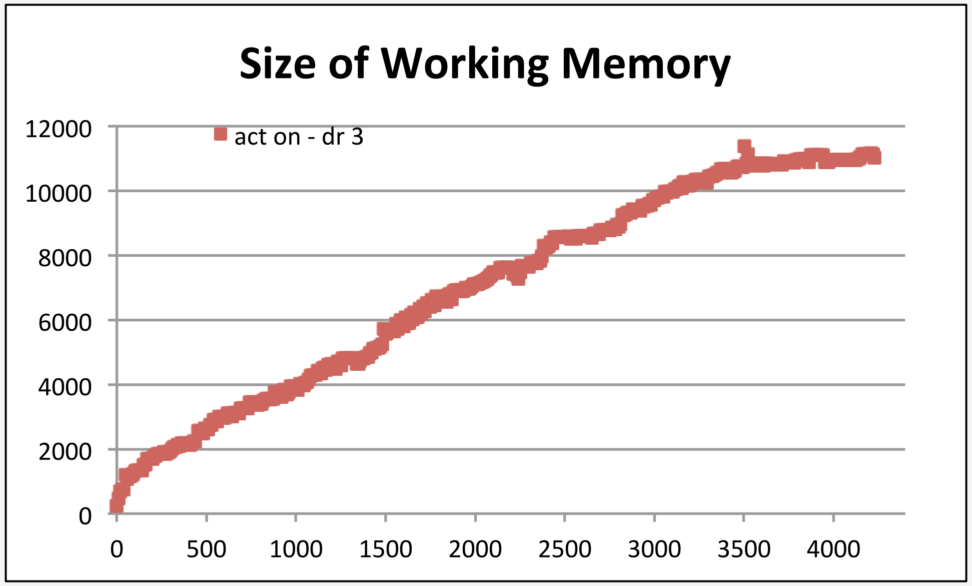
Problem #5 Hypothetical Reasoning

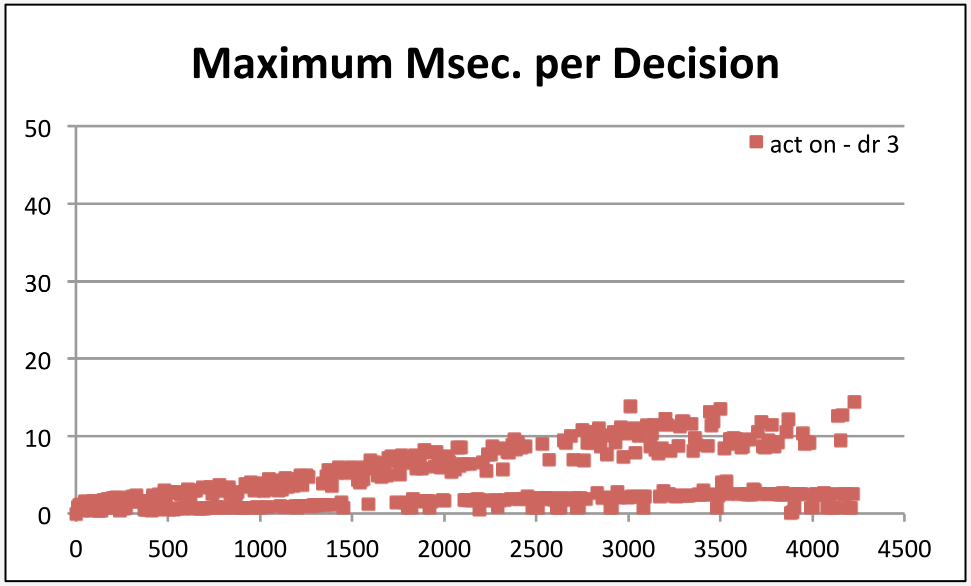
- Hypothetical reasoning could overwrite contents of SMEM if you use mirroring and aren’t careful.
  – Changes to SMEM structures in look ahead will cause them to be changed in SMEM.
  – Must copy SMEM structures if they will be modified
    - Standard in look-ahead, but might have new twists
    - No experience with this yet.

Preliminary Data on Activation in Robot Task

Decay Rate determines how fast activation decay.

3 = Very slow. Few if any removals. Similar to no decay.
6 = Fast. Sometimes has difficulty maintaining everything it needs for the problem.

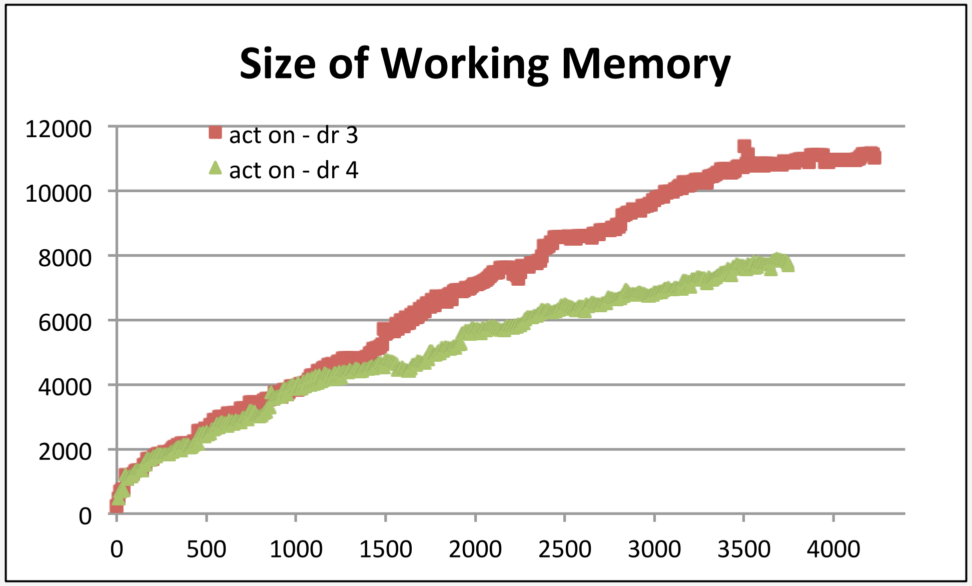


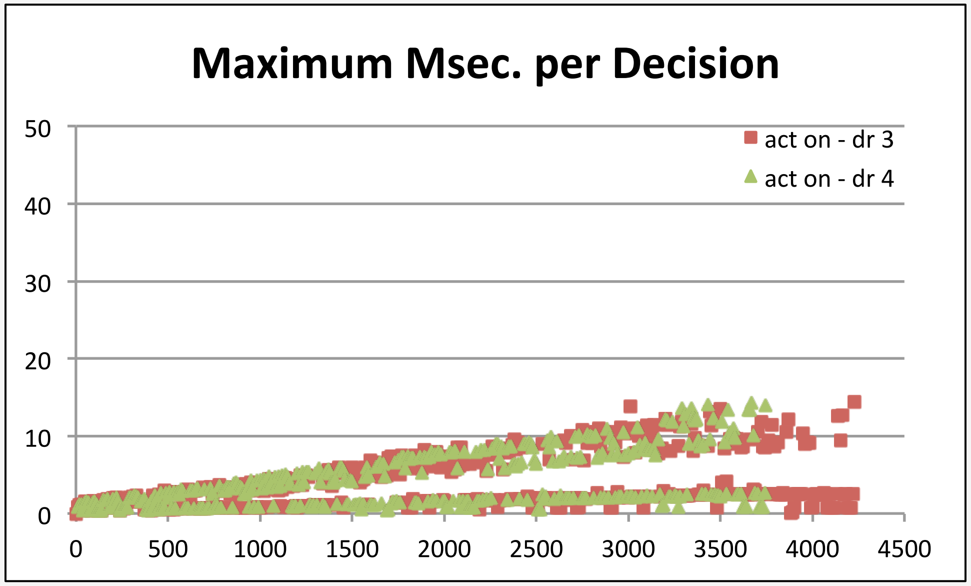


Preliminary Data on Activation in Robot Task

Decay Rate determines how fast activation decay.

3 = Very slow. Few if any removals. Similar to no decay.
6 = Fast. Sometimes has difficulty maintaining everything it needs for the problem.

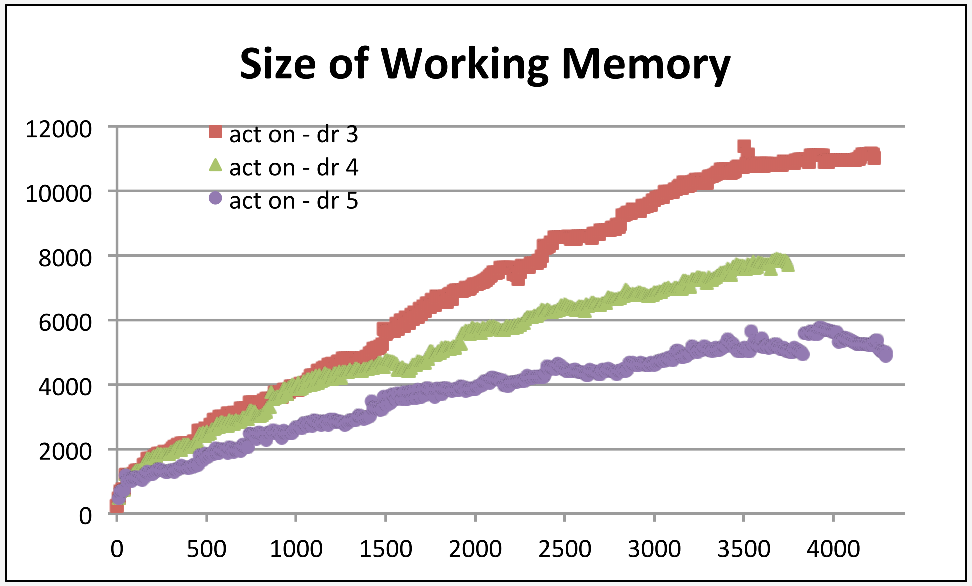


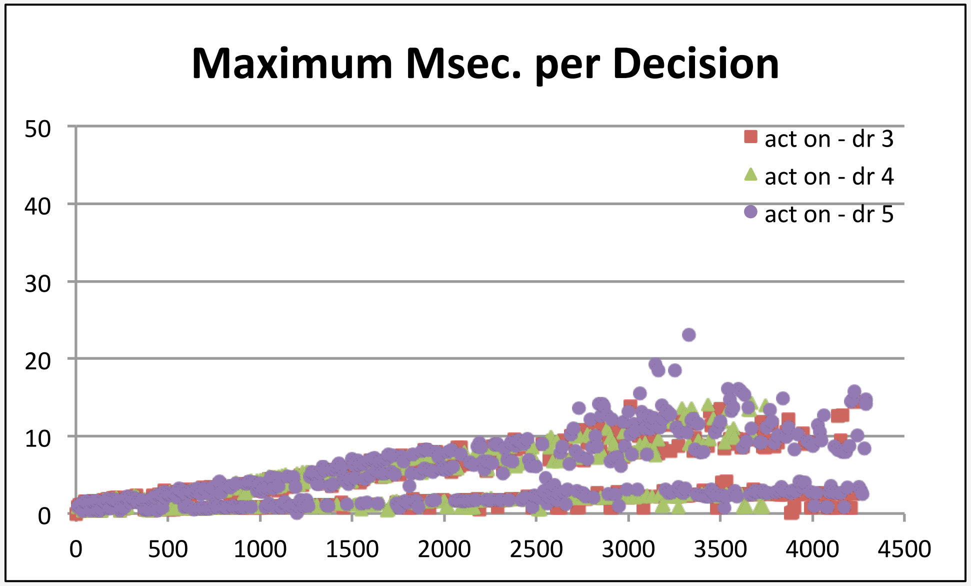


Preliminary Data on Activation in Robot Task

Decay Rate determines how fast activation decay.

3 = Very slow. Few if any removals. Similar to no decay.
6 = Fast. Sometimes has difficulty maintaining everything it needs for the problem.

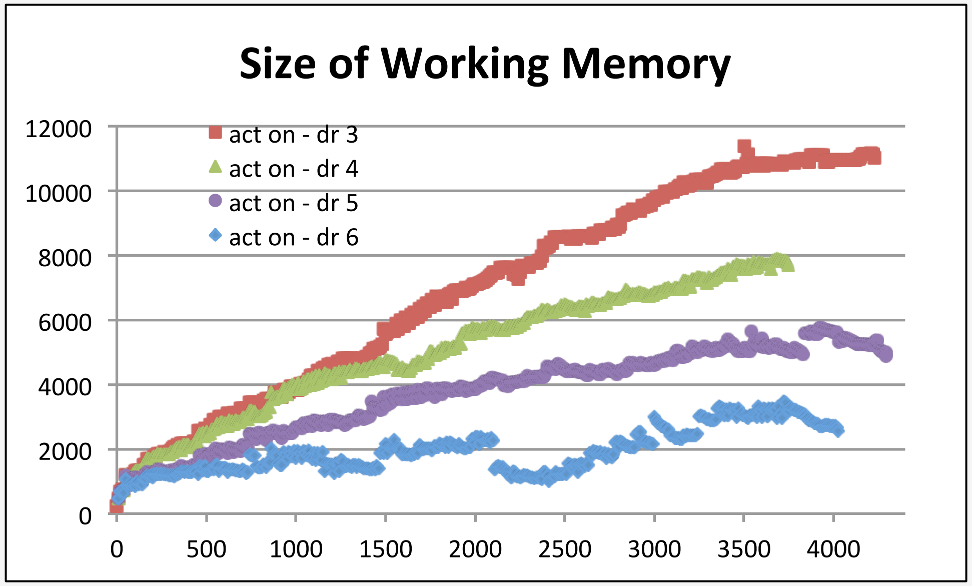


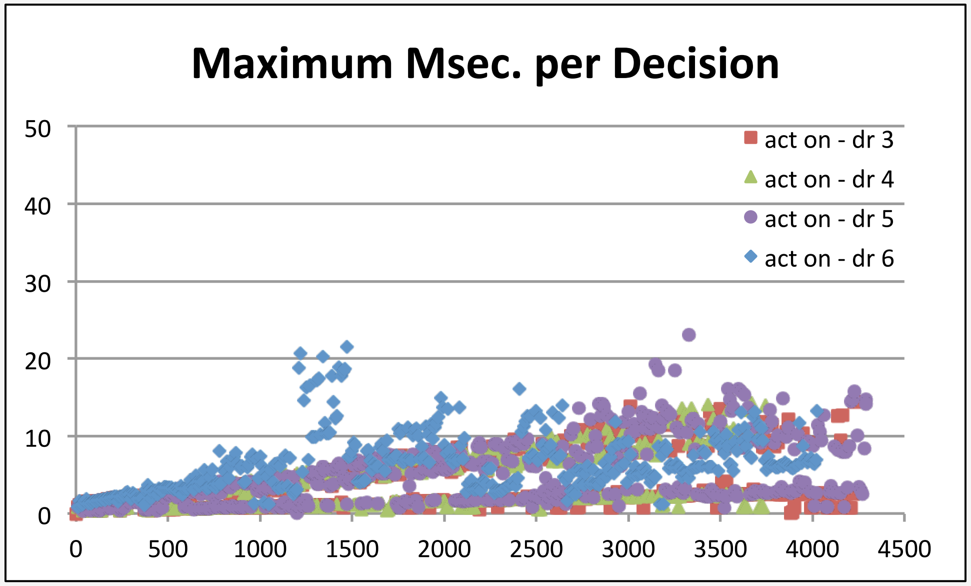


Preliminary Data on Activation in Robot Task

Decay Rate determines how fast activation decays.

3 = Very slow. Few if any removals. Similar to no decay.
6 = Fast. Sometimes has difficulty maintaining everything it needs for the problem.





Nuggets and Coal

- Takes more care when writing your agent
  - Must include special attributes (id) on smem objects.
- Creates some pressure for flat representations.
- Following a few basic rules worked for me.