Thor-Soar Agent

John Laird

Center for Integrated Cognition

https://integratedcognition.ai

May 5, 2025

Lots of help from

Nathan Glenn and James Beyer

and rest of CIC

Thor-Soar Goals

1. Create Reference Embodied Soar Agent
   - Environmental interaction: sensing, motor
   - Representations: goals, world, objects, relations, …
   - Operators: proposals & application
   - General problem and planning
   - Grounding for goal specifications and interaction
2. Platform for Agent Research
   - Language integration
     - What is the target representation for language?
   - Interactive Task Learning
     - What knowledge (content and structure) needs to be learned?

Agent Design Goals

- Flexible task/goal specification
  - Can be created from natural language interaction
- Minimal task preparation
  - Exhaustive RL is a *non-starter*
- Efficient problem solving
  - Brute-force forward search is a *non-starter*
- Efficient execution
  - Faster than real time
- Support multiple levels of world, task, & search-control knowledge

Initial Kitchen Environment for Thor-Soar

- Objects (75+)

  apple, bottle, bread, counter tops, cabinets, credit card, egg, dishwasher, drawers, floor, faucet, houseplant, light switch, lettuce, knife, kettle, microwave, paper towel roll, pan mug, pot, potato, plate, peppershaker, spatula, spoon, soap bottle, stove burner, stools, statues, sink, shelving units, tomato, toaster, vase, wine bottle, windows

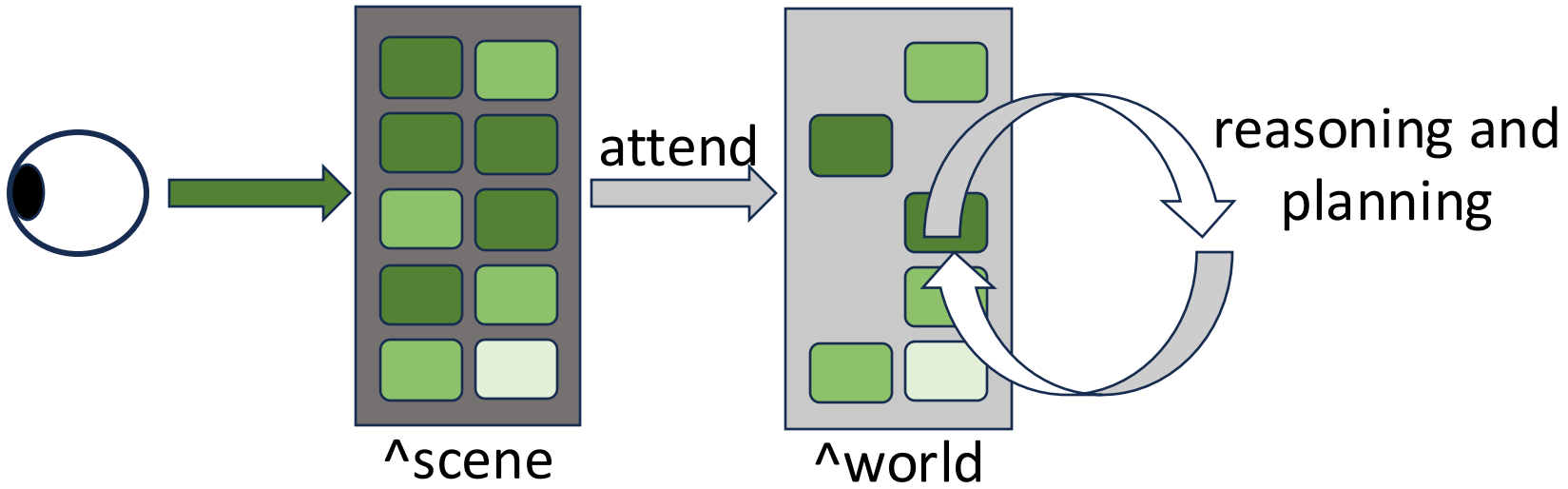
- Physical Actions (~14)

  Approach, break-object, close-container, cook, empty-object, fill-object, get-object, look, open-container, put-in, put-on, slice-object, turn, turn-off, turn-on, …

the bread is toasted;
toasted slice is on a plate and the plate is on counter

World Representation Conventions

- ^scene = all objects and relations in the current room. [~70]
  - Perceived
  - No longer perceived
  - Recalled from semantic and episodic memory
- ^world = attended objects [~5-10]
  - Objects and relations relevant to current goals and actions.
  - Makes planning and reasoning more efficient.



Agent decision making, planning, and learning >800 rules (sans NL)

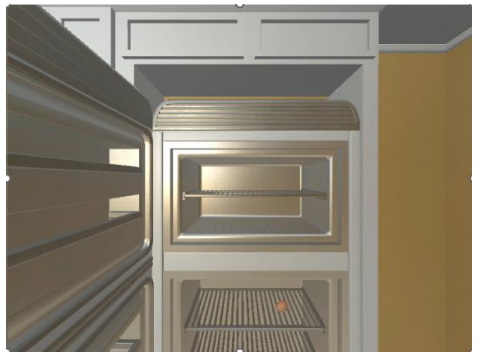
- Means-ends analysis: GPS (1959) / Prodigy (1991)
  - General search control: Priority ordering of relations & properties
    - (object position > robot location)
- Look-ahead search via iterative deepening
- Convert search to control knowledge via chunking

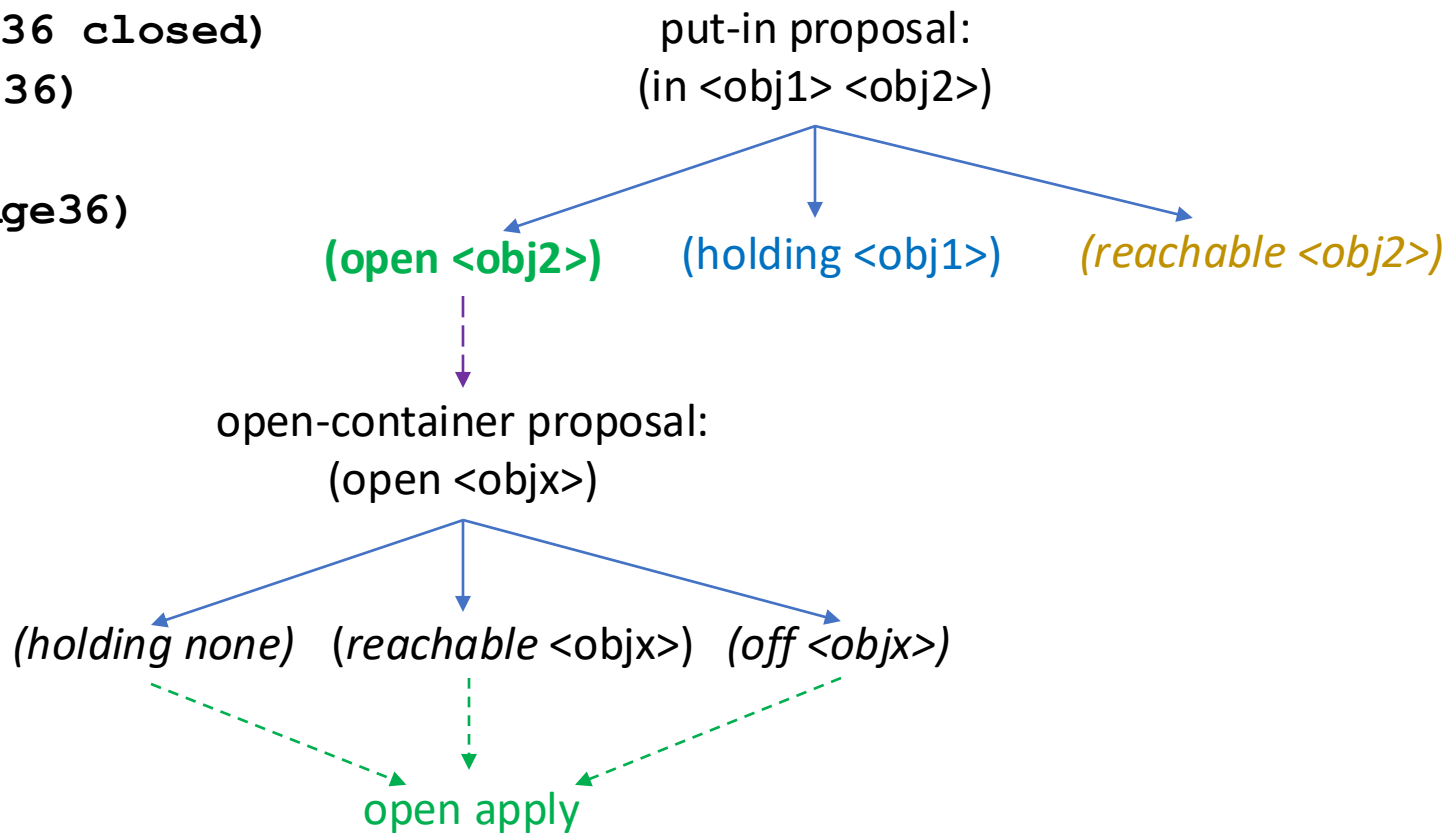
Put the potato into the fridge and close the door

- Actions:
  - Open Fridge36
  - Approach Potato50
  - Pickup Potato50
  - Approach Fridge36
  - Put-in Fridge36
  - Close Fridge36

Means-ends Analysis

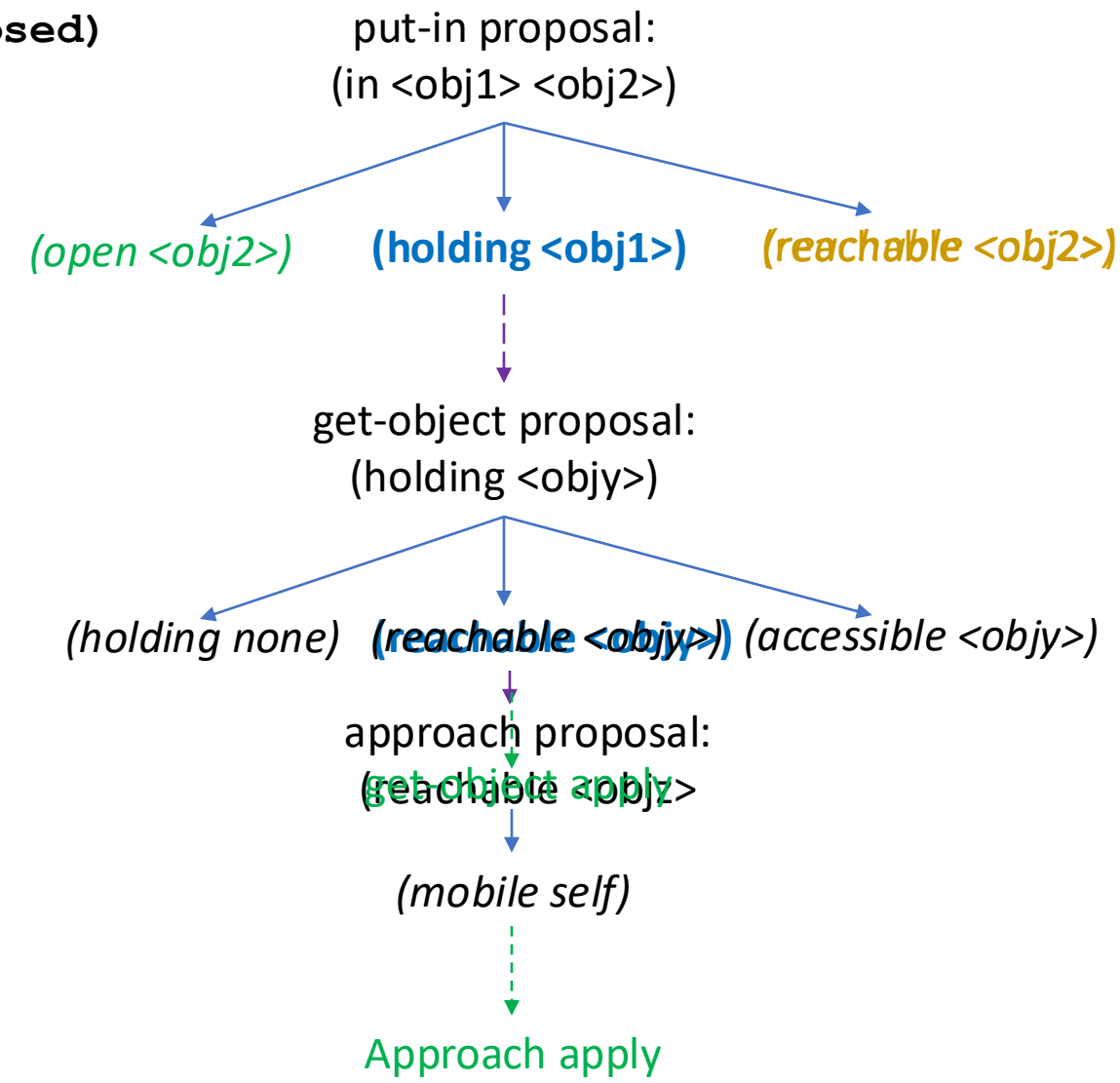
```
Goal:(in Potato50 Fridge30)  (Fridge36 closed)
1.   O: O130 (put-in Potato50 Fridge36)
2.   ==>S: S7 (operator no-change)
3.       O: O132 (open-container Fridge36)
         Apply open-container
```





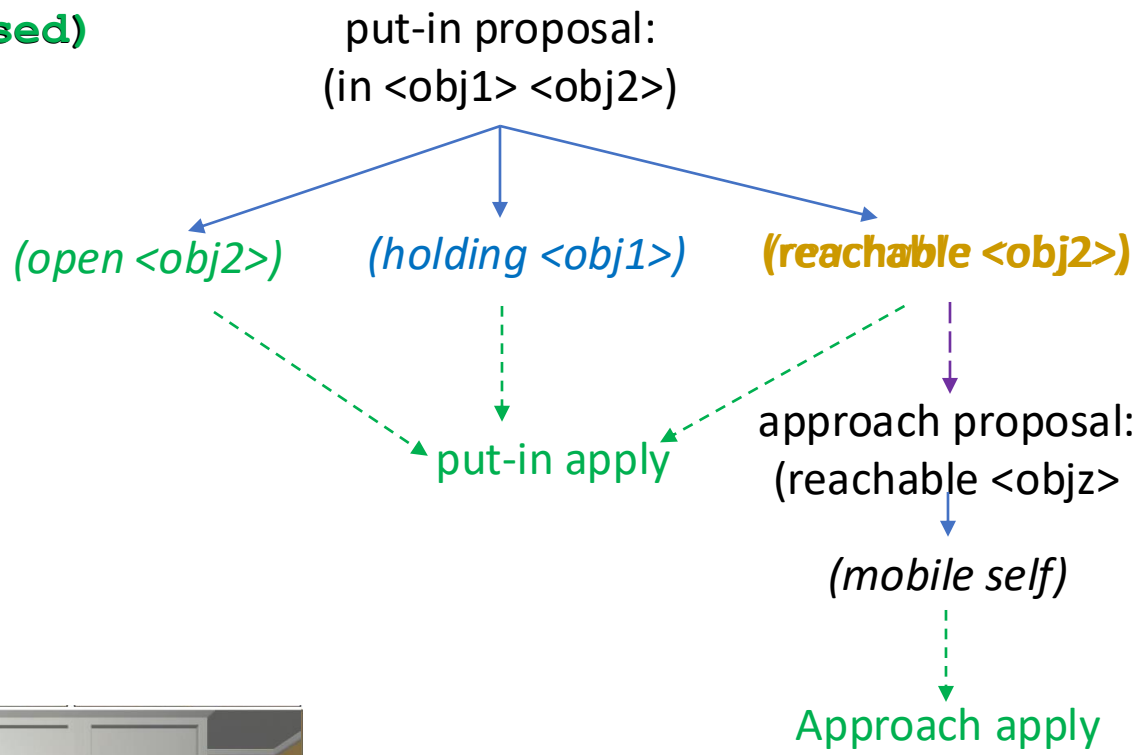
Means-ends Analysis

```
Desireds: (in Potato50 Fridge30) (Fridge36 closed)
1.   O: O130 (put-in Potato50 Fridge36)
2.   ==>S: S7 (operator no-change)
3.      O: O132 (open-container Fridge36)
        Apply open-container
4.      O: O138 (get-object Potato50)
5.      ==>S: S10 (operator no-change)
6.         O: O139 (approach Potato50)
           Apply approach Potato50
        Apply get-object Potato50
```



Means-ends Analysis

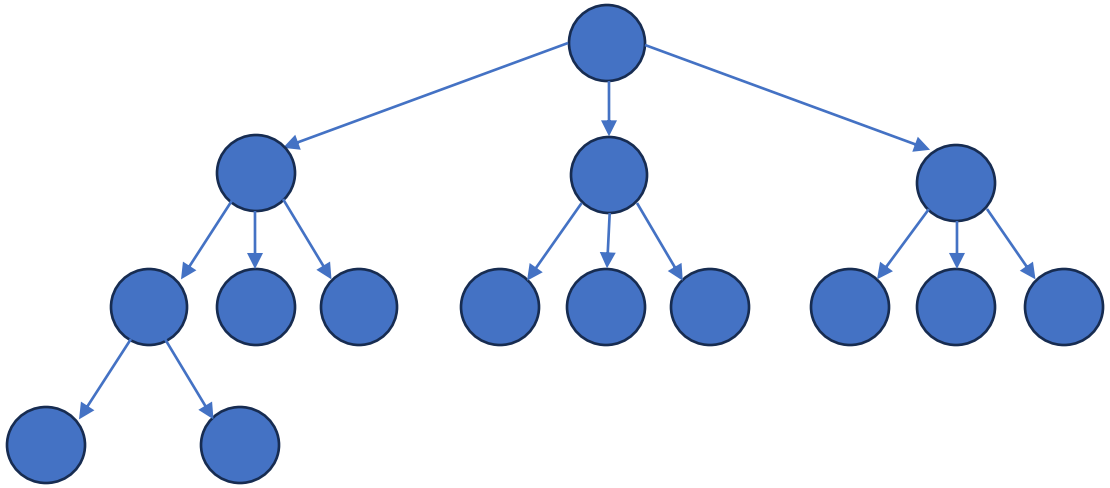
```
Desireds: (in Potato50 Fridge30) (Fridge36 closed)
1.   O: O130 (put-in Potato50 Fridge36)
2.   ==>S: S7 (operator no-change)
3.      O: O132 (open-container Fridge36)
        Apply open-container
4.      O: O138 (get-object Potato50)
5.      ==>S: S10 (operator no-change)
6.         O: O139 (approach Potato50)
           Apply approach Potato50
        Apply get-object Potato50
7.      O: O142 (approach Fridge36)
        Apply approach Fridge36
     Apply put-in Potato50 Fridge36
8.   O: O144 (close-container Fridge36)
     Apply close-container Fridge36
```



Internal Search: Iterative Deepening

- When search control knowledge is insufficient, must search
  - Multiple concurrent goals
  - Multiple actions that achieve a goal
  - Multiple objects that achieve a goal
- Iterative deepening:
  - Iteratively increase the depth of a depth-bounded depth-first search
- Compatible with means-ends analysis

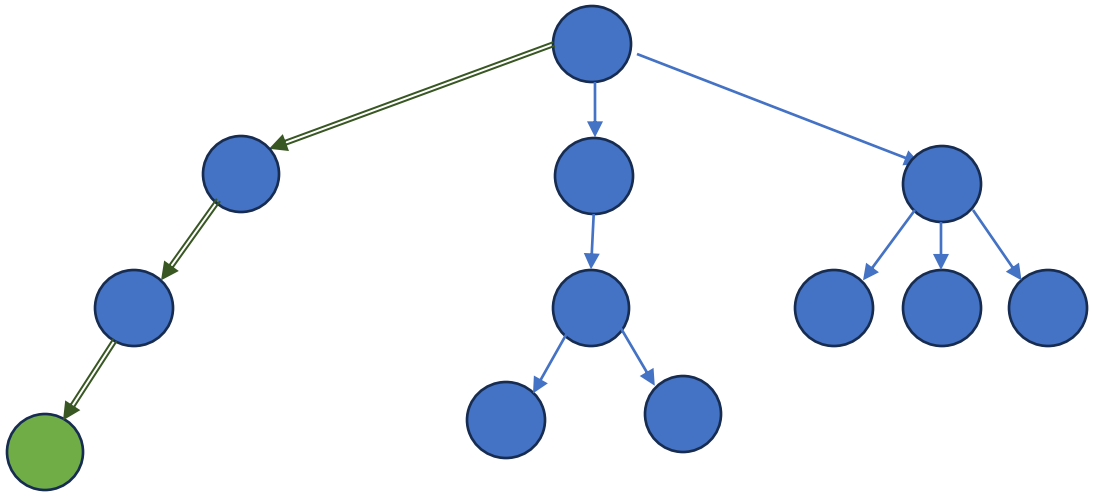
Iterative Deepening



- “make coffee” and “cook the potato” (unordered)
- The solution is at least 11 steps: 681 decisions
- But the only choice is which to select to do first
- (In Thor-Soar, cooking a potato takes longer than making coffee.)

Iterative Deepening Impasses

- Change each depth from *# of operators* to *# of impasses*
- Don’t stop until you hit an impasse



- 185 decisions
- vs. 681
- 64 after chunking

Goal Specifications

- A set of partial state descriptions of object properties and relations (desireds)
  - Can specify conjunctions, disjunctions, and temporal orderings
- NL parser translates action specifications into state descriptions
  - Put the apple in the fridge -> the apple is in the fridge.
  - Make coffee in the mug -> the mug is full of coffee
  - Allows the agent to figure out the best way to achieve the goal given its affordances
    - Issues about allowing human control over the method?
- Attempts to *ground* to an existing specific object in the world
  - Unique **^object-handle** for every object

```
(<d> ^desired.object <object>)
(<object> ^object-handle Fridge30
          ^open false)
```

Grounding Fun

- Want to use properties and relations to designate an object.
  - `The green apple on the nearby counter should be cooked.`
- An object might not currently be sensed
  - `The apple that is in the fridge should be cooked.`
- The original object is transformed to new objects with new affordances
  - Don’t know what the identity of slices will be until the bread is slice
  - `Put a slice of toasted bread on the plate.`
- Unspecified objects embedded in the operator applications.
  - `To slice an object, must be holding a knife.`
  - What knife?

Grounding: General Approach

- Create an *ungrounded* description of the object [Aaron Mininger thesis]
  - Create a unique hypothetical object-handle
  - Include that hypothetical object-handle in the goal description
- Hold *the apple* in the refrigerator

```
(<desc> ^in <in>)
(<in> ^argument1 <arg1>   #apple
      ^argument2 <arg2>)  #fridge
(<arg1> ^ungrounded true
        ^object-handle x-apple-234) # not known because in fridge
(<arg2> ^grounded true
        ^object-handle Fridge50)

(<desired> ^holding.object-handle x-apple-234)
```

Find-object Operator

- Find-object proposal:
  - an unground object used in a goal structure
- Find-object precondition:
  - an object satisfies the description of that unground object
  - becomes a goal
- Find-object action:
  - replaces the hypothetical object-handle in the goal structure

```
(<object> ^object-handle x-apple-234 -
                                Apple3)
```

Knowledge Operators

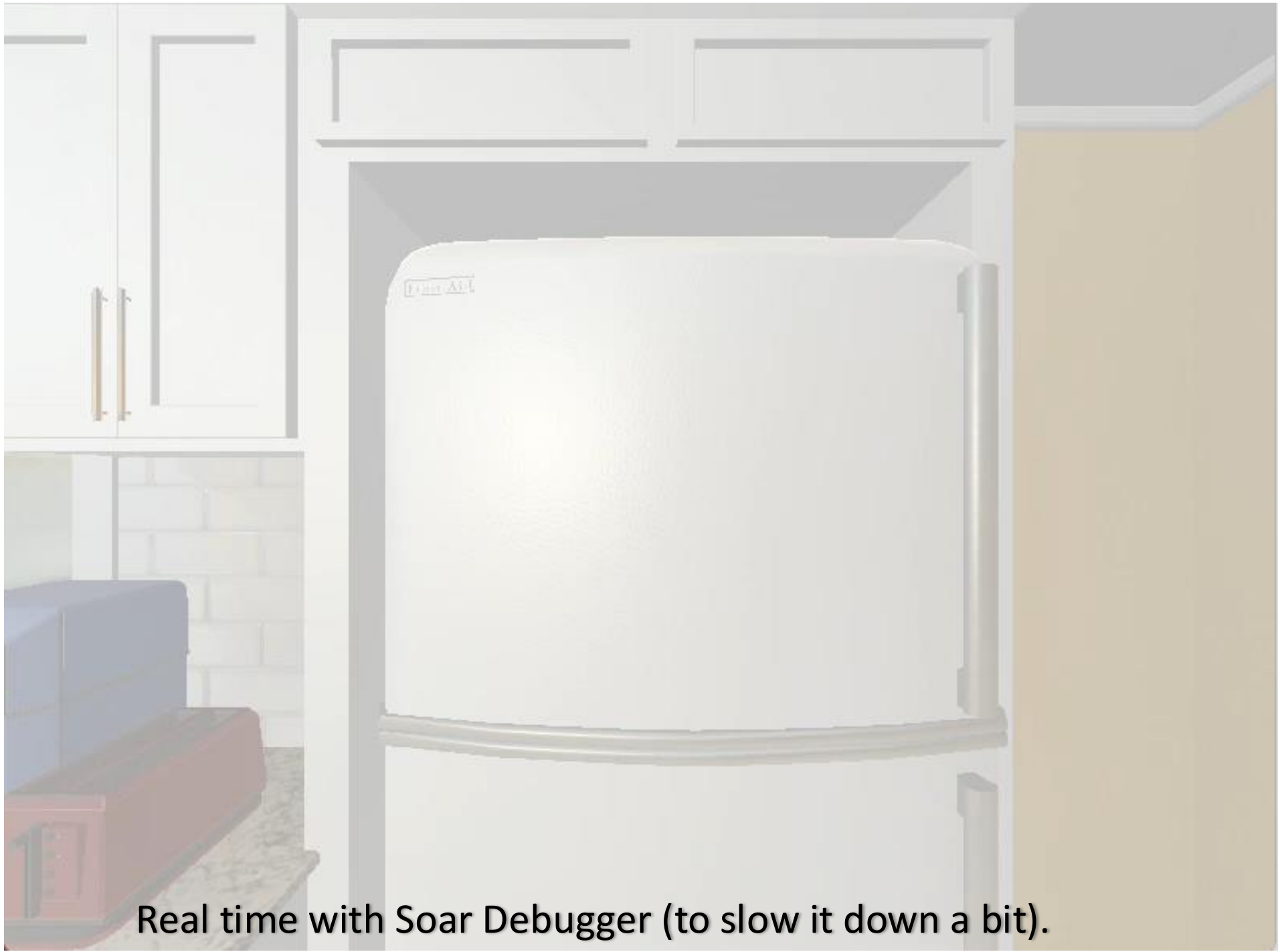
- Search for an object that satisfies the description
  - Ground-to-scene-object
    - An object in the scene matches the description
  - Search-semantic-memory
    - Query Smem to find a known object that matches the description
  - Search SVS
  - Search episodic memory
    - Lots of promise!
  - Explore-the-world
    - Open cupboards, go to other rooms, …
  - Ask a human
- MEA and iterative deepening can direct the search

Remove Omniscient Sensing: Mostly Implemented

- Limit sensing to a viewing frustum
- Scan area and explore
  - Explore containers and remember contents
- Store object information in semantic memory
  - Recall when re-entering room
- Learn affordances through experimentation
  - Openable, grabbable, fillable, cleanable
  - *Not breaking, slicing, or cooking!*
  - Associate with object category and store in semantic memory

- Defines agent conventions
- General and efficient problem solving, search, and execution
- Integrates MEA and iterative deepening and chunking
- Initial integration with language

- Still many rough edges
  - Some overly specific search control
  - Some chunking issues
  - Some grounding issues
- Need to exploit episodic memory
- Need to create a domain-independent version of the code: new default
  - Have applied to Blocks World
- Issues with intertwined MEA subgoals



Real time with Soar Debugger (to slow it down a bit).

Goal Specification Flexibility: Dynamic Grounding

- Initially assume goal specification was grounded to a specific object in the world.
  - Unique object-handle for every object and in included in goal
  - **`(in Potato50 Fridge30) (Fridg36 closed)`**
- Distinguish two different specifications
  - Desired object properties/relations *to be achieved*:
    - **`The potato should be in the fridge.`**
  - Properties/relations that determine the desired object in the original situation.
    - **`Get the apple that is in the fridge.`** - That apple might never have been seen.
- Tie together descriptions with an invented object-handle and label object description as *ungrounded*

- Assume ultimately have to ground a specification of an object to a specific object.
- Distinction between properties of the desired object (cooked, etc.) and properties of the object as it exists in the world (location, …).
- Specify properties, not an id of desired.
  - Category, …
  - *Get the egg that is in the refrigerator: get-egg-in-fridge*
- Handle objects that are destroyed and change id
  - Slicing, breaking
  - What object it will have been derived from
- Internal desired object specification
  - Need to find a place to put down an object
  - Need to find an object with certain properties

Agent Development: ~820 rules

- Initially assume lots of knowledge: omniscient sensing, object affordances, actions
- Agent decision making and planning
  - Means-ends analysis
    - Select operator that achieves a desired
    - Unachieved preconditions become new desired
    - *Not just task operators (knowledge operators)*
  - Control knowledge
    - Difference priority (in = on > open = closed = accessible > holding > reachable > activated)
    - Precondition preference: Get X: holding none > accessible X > reachable X
  - Iterative deepening look-ahead search:
    - Shortest execution time
    - Intermixes actions with different durations
  - Chunking
    - Learns selection knowledge
    - Learns direct execution (some issues)
    - No extensive cumulative chunking experiments
- No reinforcement learning or episodic memory use cases yet