How to Build a Soar Agent

John Laird

John L. Tishman Professor of Engineering

Division of Computer Science and Engineering

University of Michigan

Overview

- Describe my approach to building Soar agents
- Use the clean-room bot as example
- Will make source code available in week or two
  - Needs to be cleaned up

Goals of Agent Design

- Agent interacts with environment in real time.
- Focus on immediate goals.
  - Start with bot that has one goal
  - Generalize to one that can have multiple missions
- Agent has potential to use all of Soar capabilities.
- Start with storage in working memory

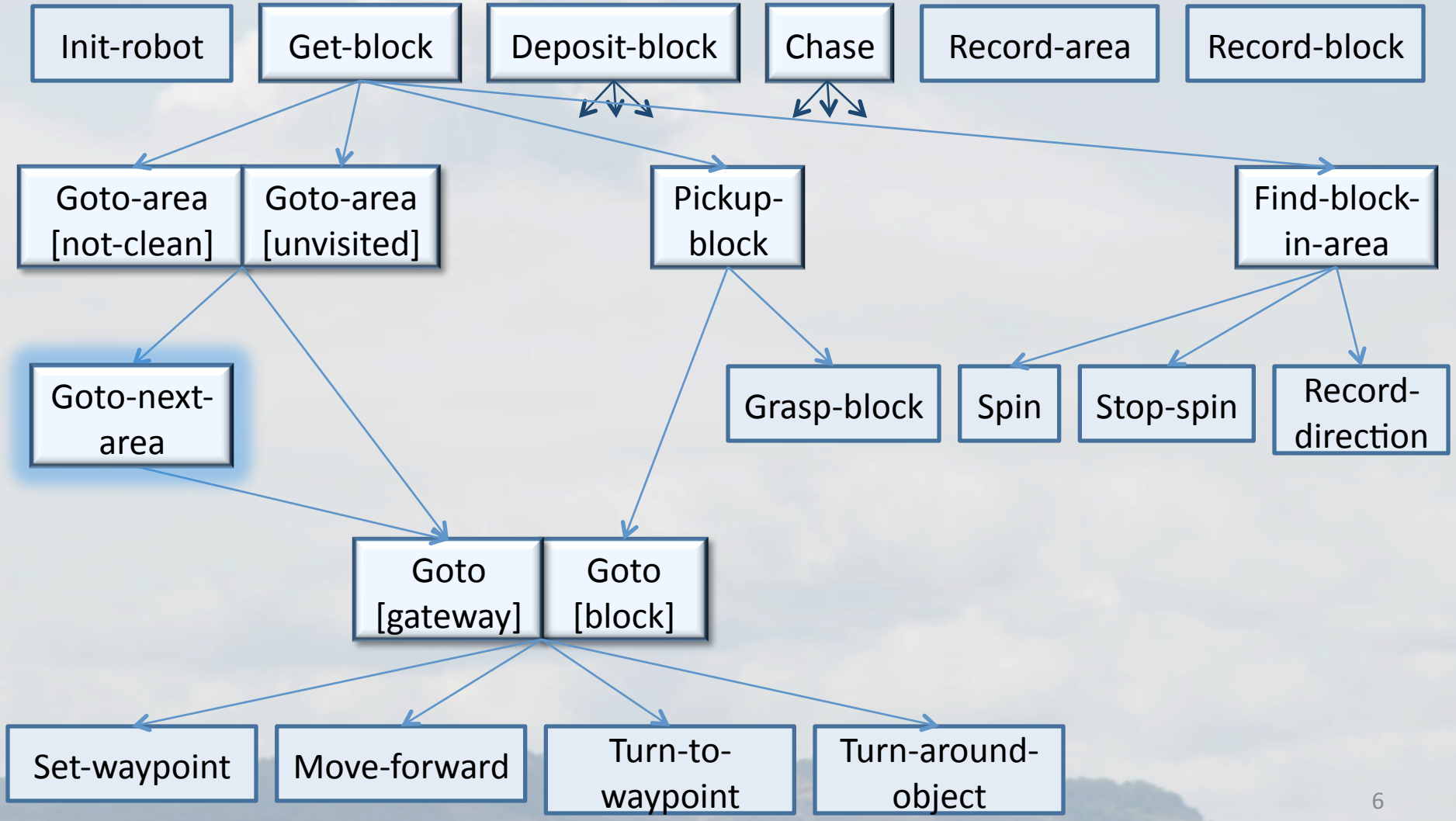
Clean-room Agent

- Explores household looking for blocks
  - Builds up map as it explores
  - Map consists of rooms and doorways connected in a graph structure
- Takes blocks to storage room
  - Searches for shortest path.
- Chases mice it sees while it is cleaning house

Overall Design Ideas

1. Use operator no-changes to create immediate task hierarchy.
2. Maintain mission structure in wmem of smem as linked list (or tree).
3. Movement is deep in operator hierarchy.
4. Store intermediate progress on top-state (GDS!)
5. Rely on reinforcement learning to learn control knowledge when possible.
6. Use selection space for planning.
7. Imagery is not yet ready for prime time.

Overall Structure of Agent



Trace

```
  1: O: O3 (init-robot)
  2: ==>S: S6 (operator no-change)
  3: O: O5 (record-area)
  4: O: O2 (get-block)
  5: ==>S: S8 (operator no-change)
  6:    O: O7 (goto-not-clean-area)
  7:    ==>S: S10 (operator no-change)
  8:       O: O9 (go-to)
  9:       ==>S: S12 (operator no-change)
 10:          O: O10 (set-waypoint)
Add waypoint: 14
Rotating to waypoint at heading 13
 11:          O: O12 (turn-to-waypoint)
 12:          ==>S: S15 (operator no-change)
 13:             ==>S: S17 (state no-change)
 14:                O: O13 (wait)
 16:                O: O14 (wait)
...
 59:          O: O58 (move-forward)
 60:          ==>S: S20 (operator no-change)
 61:             ==>S: S22 (state no-change)
...
Rotating to waypoint at heading 18
338:          O: O336 (turn-to-waypoint)
339:          ==>S: S25 (operator no-change)
340:             ==>S: S27 (state no-change)
```

Init-task

- Define task parameters
- If mission – initializes mission

Record-x

- Have top-state operator to record important perceptual features: objects, rooms, …
  - Store persistent information in wmem or smem.
  - Rely on episodic memory for object locations.
- Preferred over task operators
  - Blow away substate stack to record information
  - Avoids issues with chunking
  - Substate stack regenerates

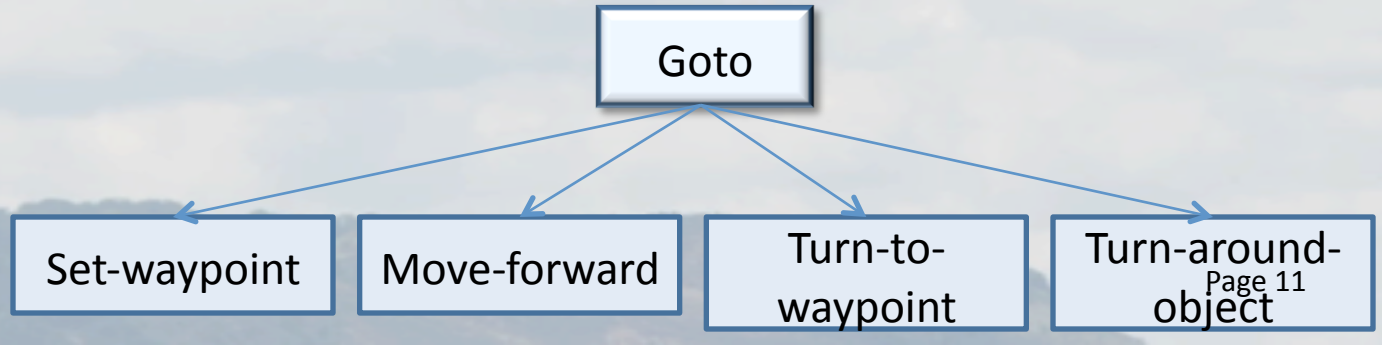
Parameters

Maintain run-time parameters in working memory, created by an elaboration:

```
sp {robot*elaborate*state
   (state <s> ^name robot
              ^superstate nil)
-->
   (<s> ^parameters <p>)
   (<p> ^pickup-range 1.0    # range in meters
            ^collision-range 1.0
            ^low-linear-velocity .3
            ^medium-linear-velocity .6
            ^high-linear-velocity .7
            ^low-angular-velocity 5
            ^medium-angular-velocity 40
            ^high-angular-velocity 60
            ^waypoint-tolerance 0.3
            ^message-update-time 3
            ^rotation-tolerance 8     # degrees the agent's yaw can be off
            ^range-tolerance 1        # distance at which arc is considered blocked
            ^progress-update-time 20  # how often progress is updated
            ^max-angular-distance 10
            ^min-angular-distance -10
            ^areas-held-in wm
            ^objects-held-in wm
            ^look-ahead-planning yes
            ^search-control-go-to-gateway yes
            ^default-storage-area-id 9
            ^increased-angular-tolerance-range .5
            ^increased-angular-tolerance-multiplier )}
```

Use of Elaborations

- Compute qualitative movement features:
  - yaw-aligned
    - is agent heading in right direction given tolerances?
  - destination-in-range
    - in range of destination?
- Test for absence of incorrect values
  - avoids blinking



Follow the Rules

- Don’t mix PSCM functions in a single rule
- Don’t fight the GDS
  - it is there to keep you honest!
- Get to know the default rules
  - planning (and chunking) can be fun!
- Don’t get too clever (keep it simple …)
  - Don’t try to avoid operators, substates, etc.
- Avoid maintaining large sets in working memory