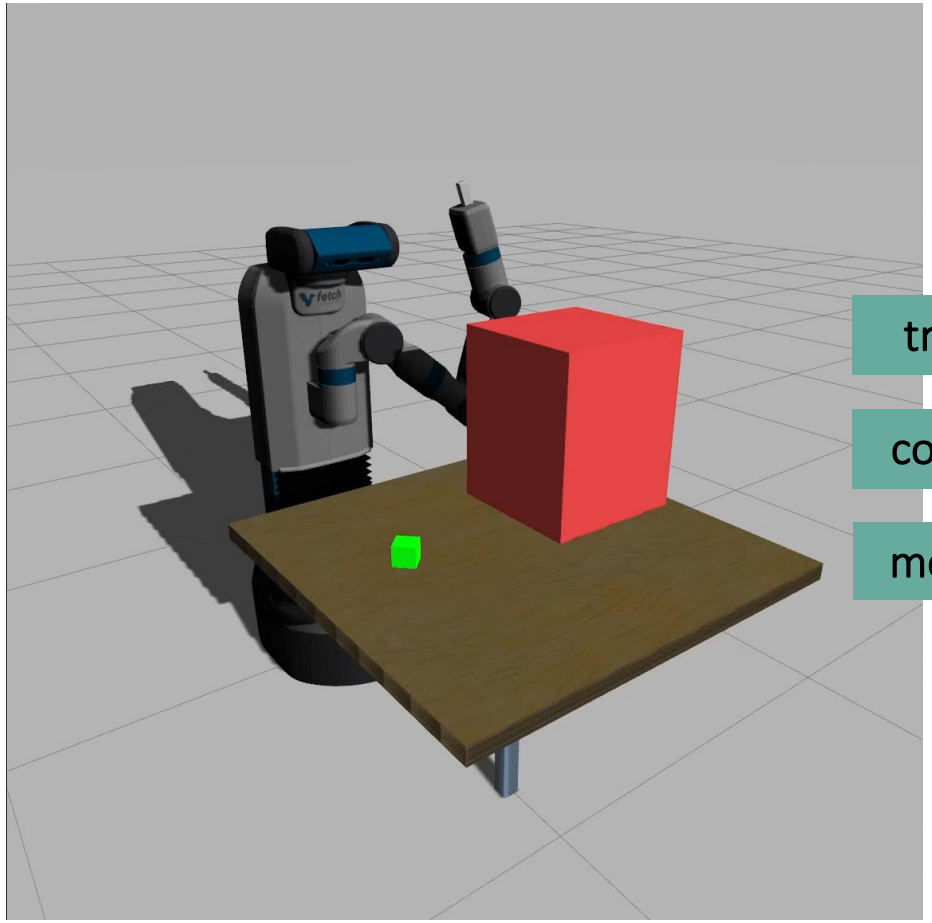
Flexible motion control through SVS

New features and example agents

Soar Workshop 2024 • Lizzie Goeddel

Standard Soar motion control

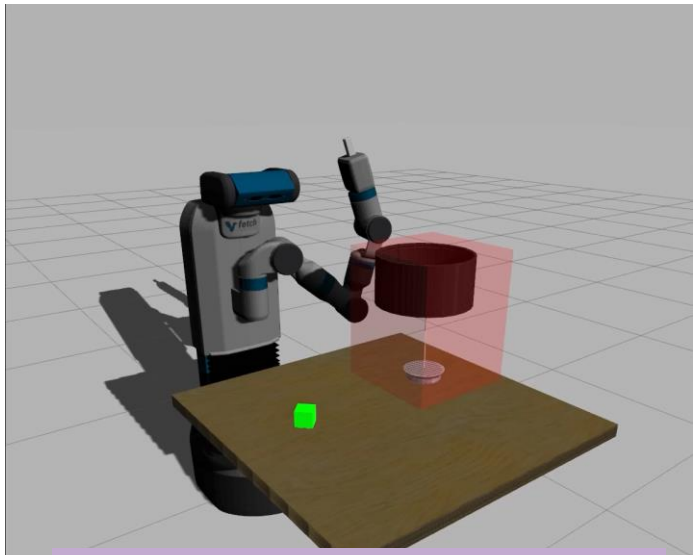
```
(S1 ^io I1 ^svs V1)
 (I1 ^input-link I2
     ^output-link I3)

                          )

              finished)
 (V1 ^command C3
     ^spatial-scene I4)
  (I4 ^id world ^child C5
     ^child C6 [...])
   (C5 ^id target_cube)
   (C6 ^id generic_obstacle)
```



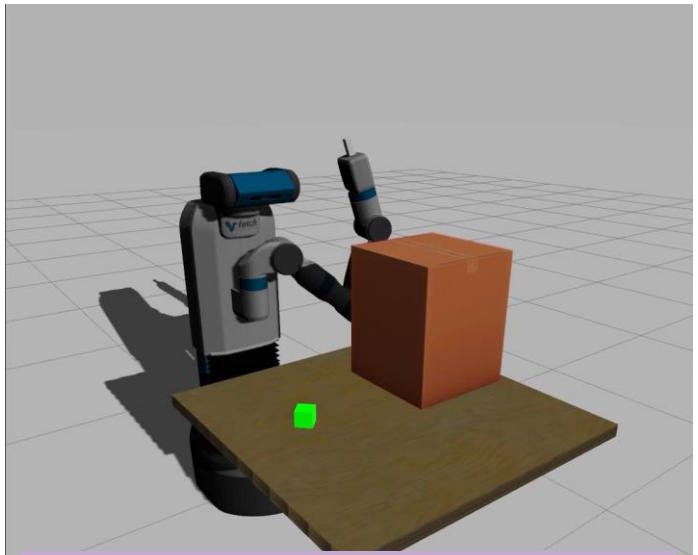
trajectory

continuous

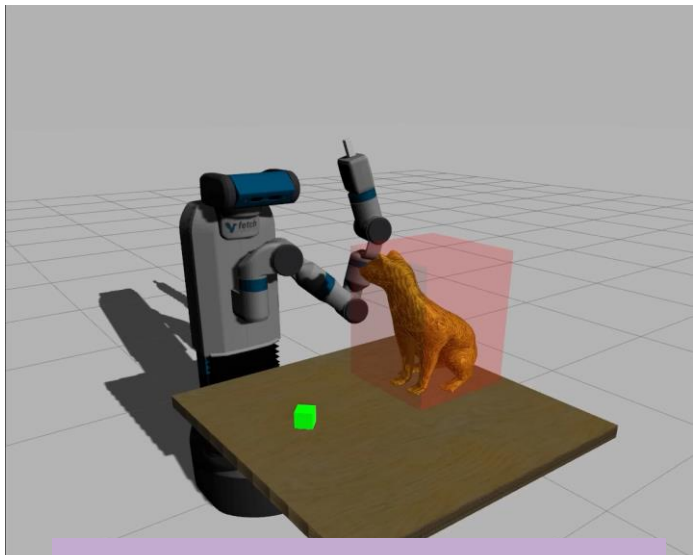
modality-specific



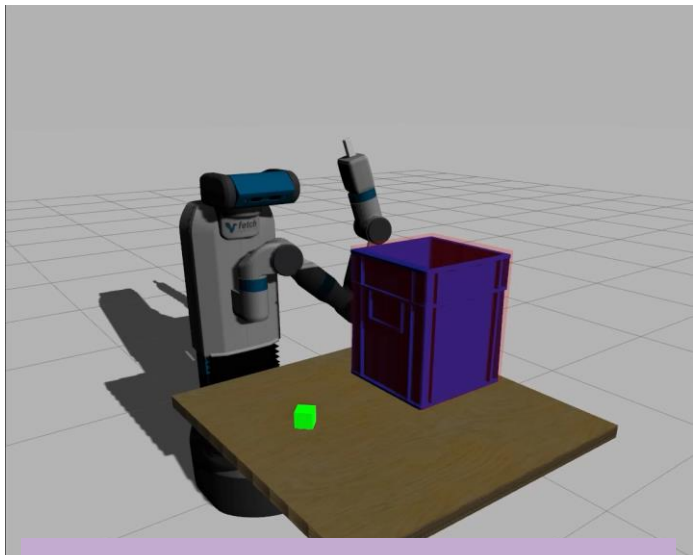
```
(C6 ^id lamp
    ^property fragile)
```



```
(C6 ^id cardboard_box
    ^property hittable)
```



```
(C6 ^id cat_statue
    ^property moving)
```

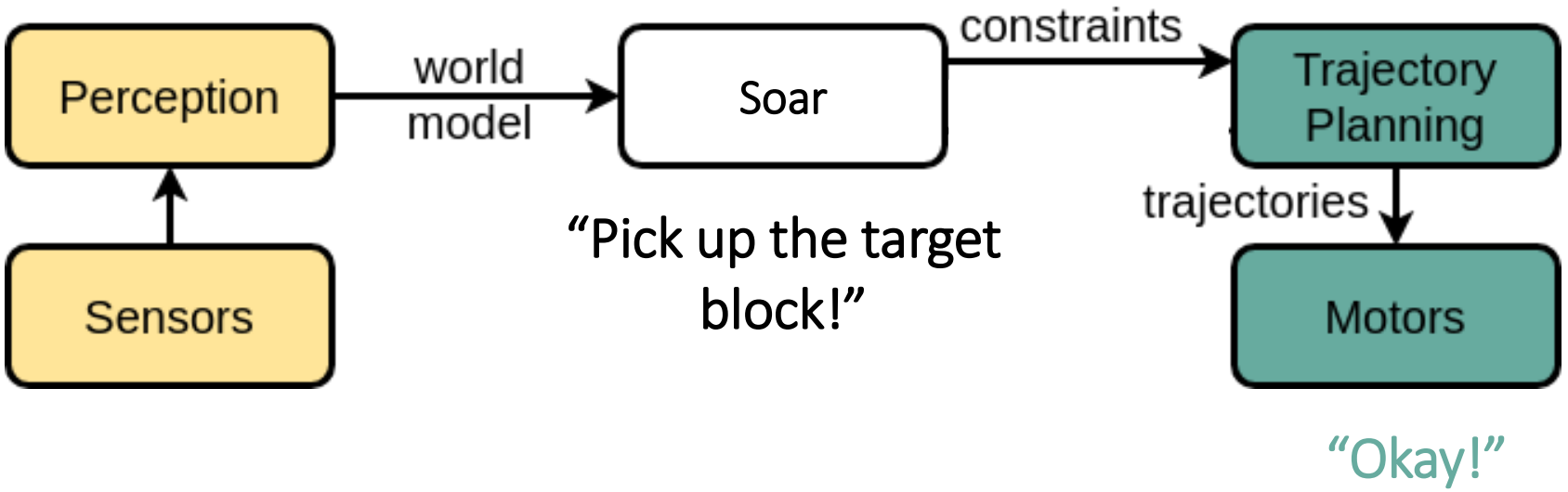


```
(C6 ^id dropoff_box
    ^property container)
```

What if the obstacle's identity changes?

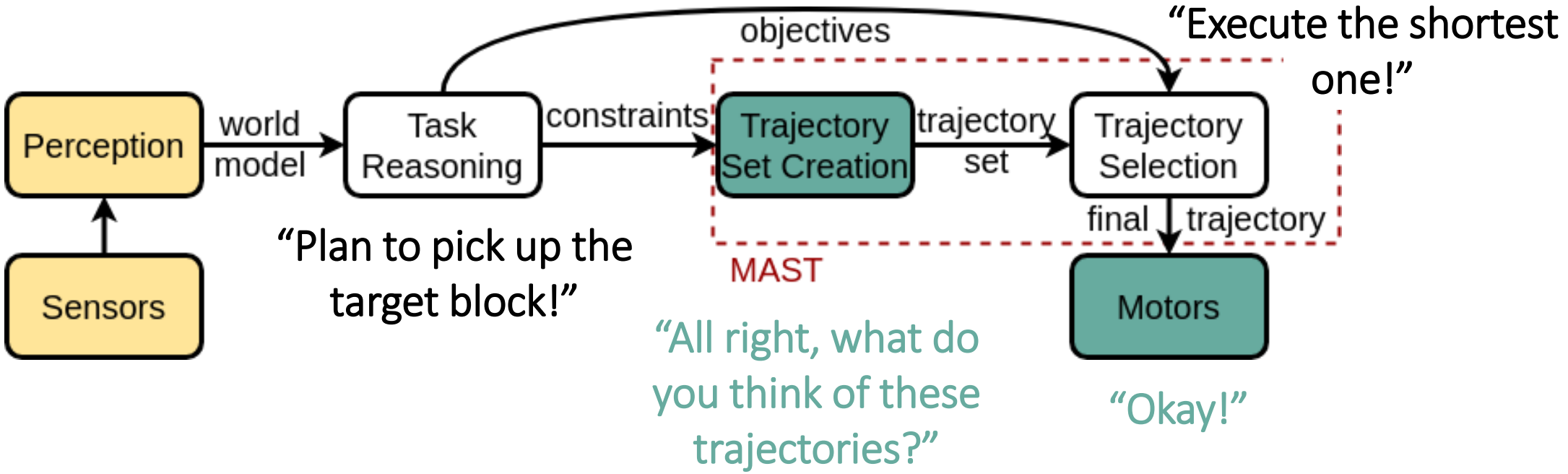
Standard output-link interface provides **no mechanism** to adjust motion trajectory accordingly

Problem: Subsystem-independence

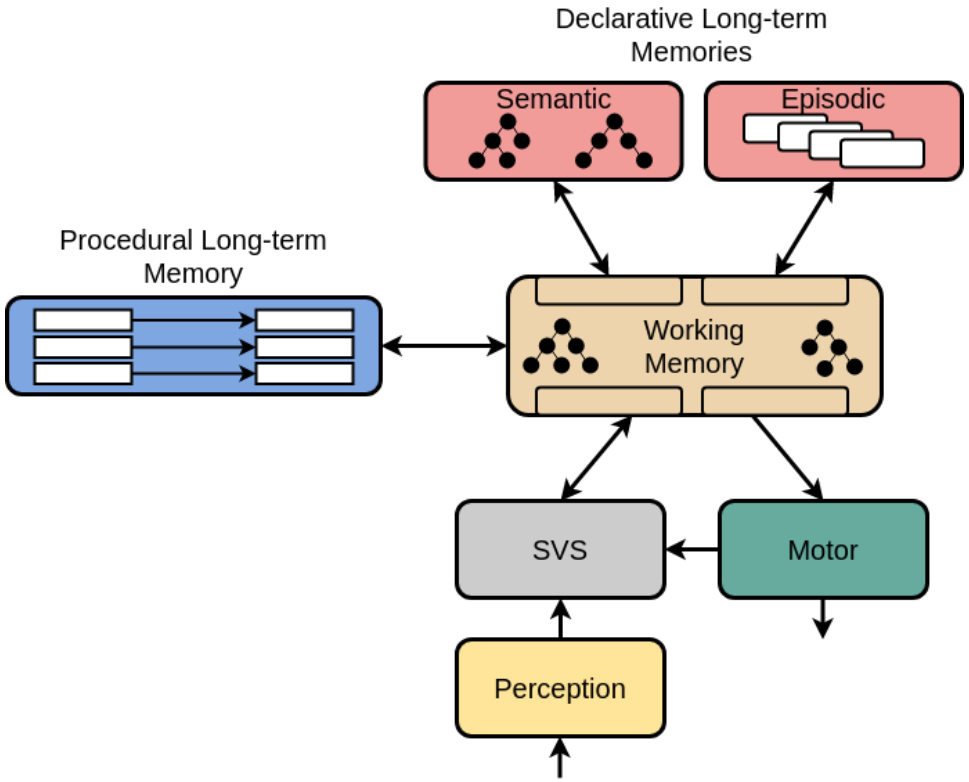


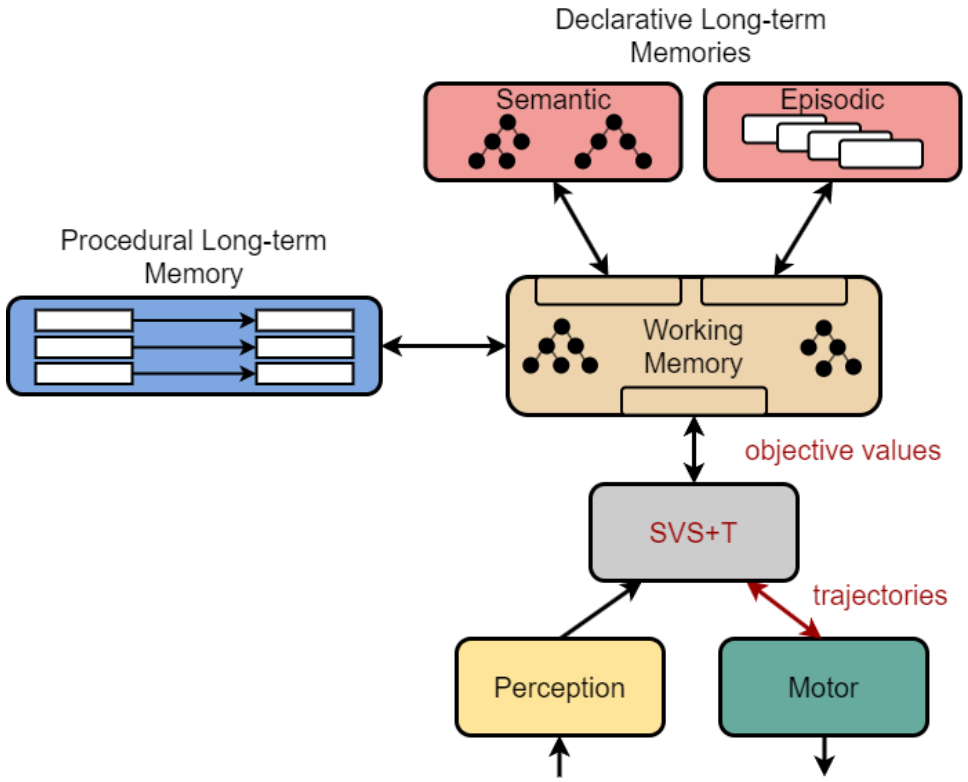
Solution: MAST

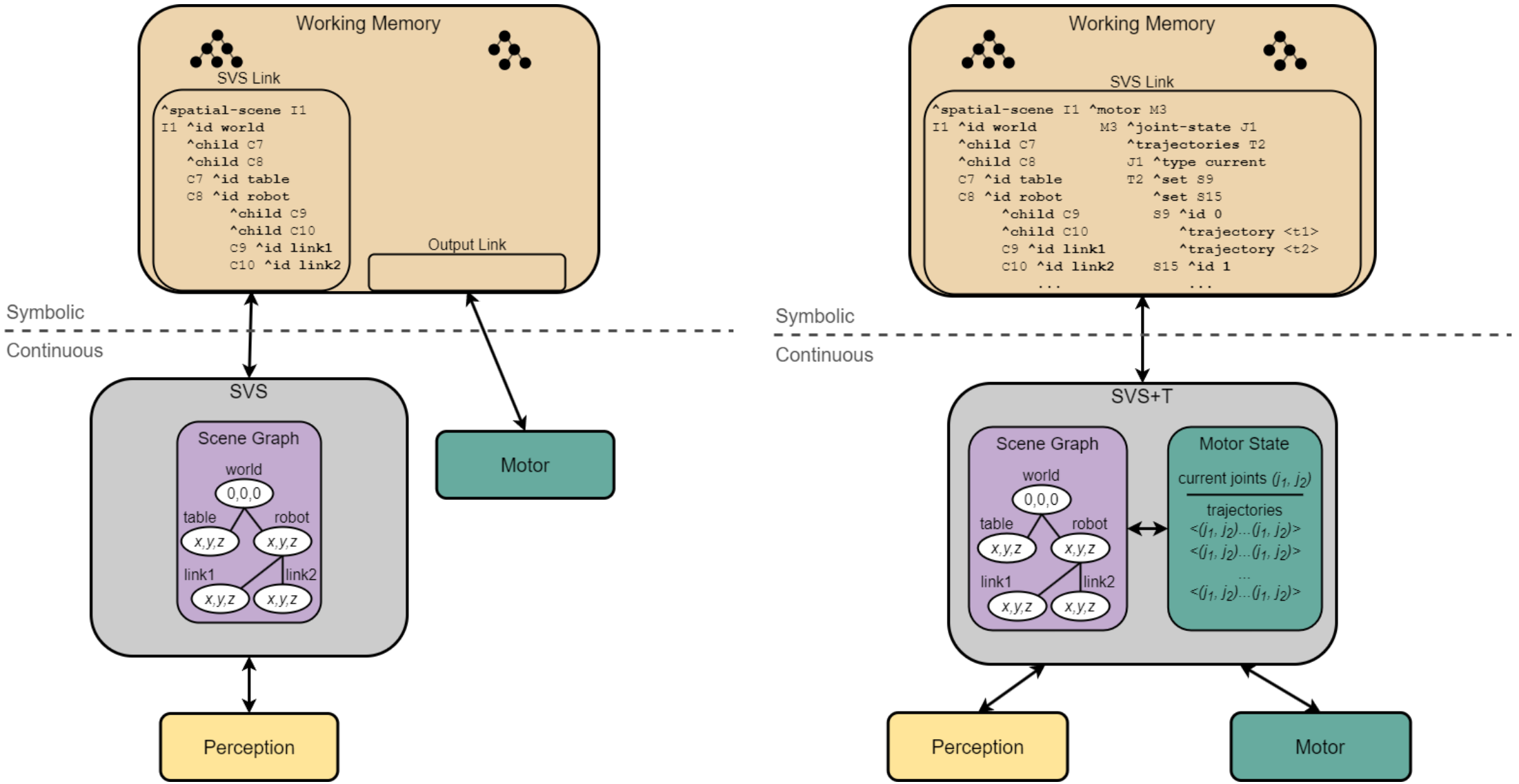
Motion planning with **A**gent **S**election of **T**rajectory



Soar + MAST = SVS+T






Working Memory
SVS Link
^spatial-scene I1
I1 ^id world
^child C7
^child C8
C7 ^id table
C8 ^id robot
^child C9
^child C10
C9 ^id link1
C10 ^id link2
Output Link
Symbolic
Continuous
SVS
Scene Graph
world
0,0,0
table
robot
x,y,z
x,y,z
link1
link2
x,y,z
x,y,z
Motor
Perception
Working Memory
SVS Link
^spatial-scene I1 ^motor M3
I1 ^id world
M3 ^joint-state J1
^child C7
^trajectories T2
^child C8
J1 ^type current
C7 ^id table
T2 ^set S9
C8 ^id robot
^set S15
^child C9
S9 ^id 0
^child C10
^trajectory <t1>
C9 ^id link1
^trajectory <t2>
C10 ^id link2
S15 ^id 1
Symbolic
Continuous
SVS+T
Scene Graph
world
0,0,0
table
robot
x,y,z
x,y,z
link1
link2
x,y,z
x,y,z
Motor State
current joints (j1, j2)
trajectories
<(j1, j2)...(j1, j2)>
<(j1, j2)...(j1, j2)>
...
<(j1, j2)...(j1, j2)>
Perception
Motor

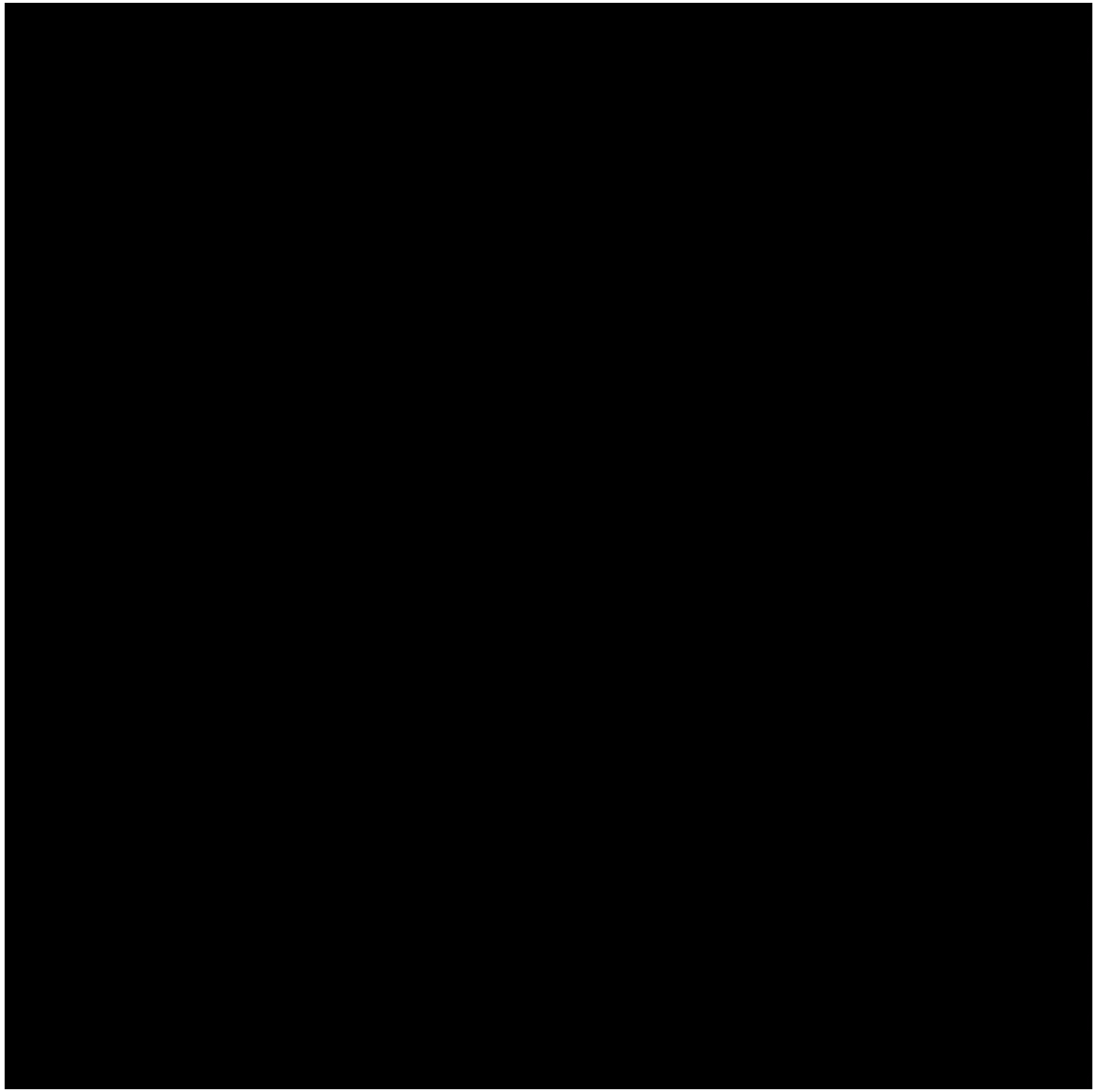
SVS+T motion control

```
(S1 ^svs V1)
 (V1 ^command C3 ^motor M1)
  (C3 ^find-trajectories F1)
   (F1 ^max-number 30 ^target T3)
       ^id 0 ^status finished)
  (C3 ^evaluate-trajectories E2)
   (E2 ^set-id 0 ^type select
       ^objective execution-time)
       ^status success)
  (C3 ^execute-trajectory E3)
   (E3 ^set-id 0 ^trajectory-id 2)
```

```
(M1 ^joint-state J1 ^trajectories T1)
 (T1 ^set I10)
 (I10 ^command-id 0)
      ^trajectory T5
      ^trajectory T6
      ^trajectory T7
           …
      ^trajectory T34)
   (T5 ^id 0)
   (T6 ^id 1)
   (T7 ^id 2)^selected-by execution-time)
        …
   (T34 ^id 29)
```

DEMO

Oh great, this works!



Agent development

To pick up:

```
(O4 ^name pick-up
    ^object-id target_block
    ^objectives O5)
```

To put down:

```
(O8 ^name put-down
    ^object-id can_coke
    ^target-id grey_tray
    ^objectives O9)
```

To select only:

```
(O12 ^name do-selection
     ^trajectory-set 1
     ^objectives O15)
```

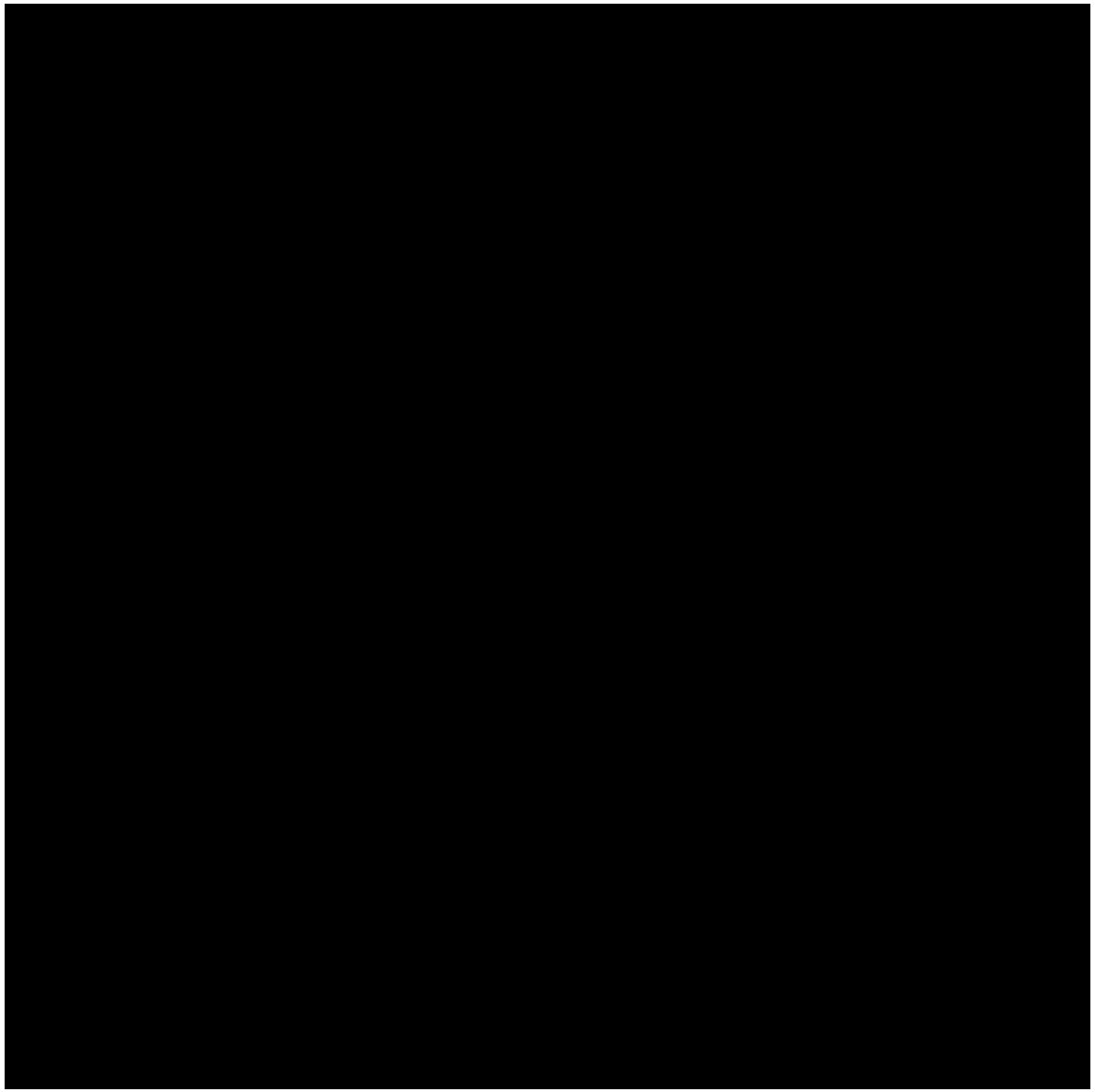
An objective list:

```
(O8 ^first F3
    ^second E5
    ^third T8)
  (F3 ^name end-effector-length
      ^parameters P2)
    (P2 ^direction min)
  (E5 ^name end-effector-rotation
      ^parameters P3)
    (P3 ^direction min)
  (T8 ^name min-clear-subset
      ^parameters P4)
    (P4 ^direction max
        ^obstacle lamp)
```

Objective table from Soar command line: `svs objectives`

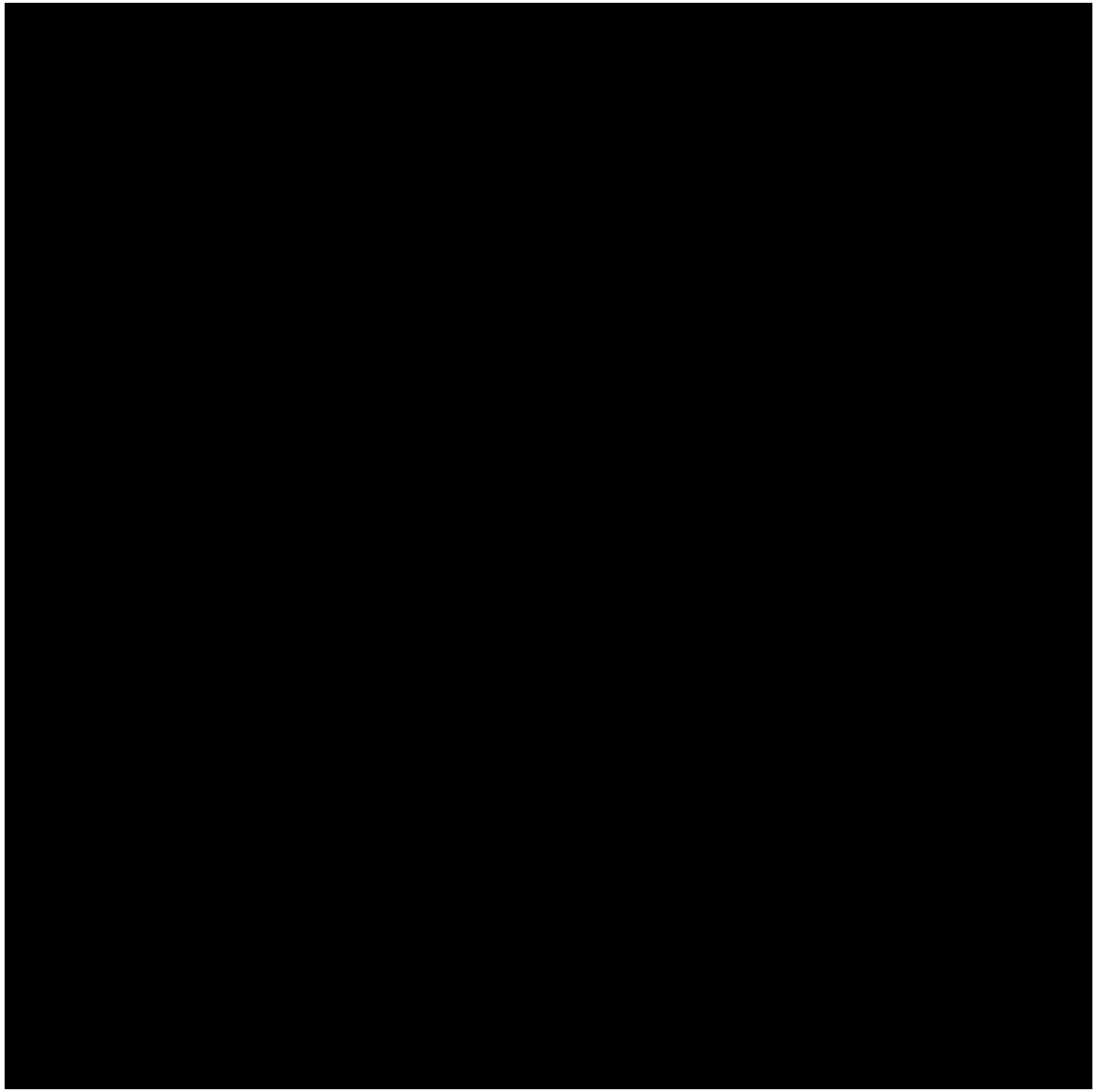
DEMO

New agent skills!



DEMO

Agent skills matter!



Nuggets and coal

+ New agent abilities
+ New objectives to use
+ Default motion substates
+ Evaluations validate expectations
+ Awesome demos!
+ Defense in July!

- Not in Soar releases
- A bit more evaluation to do