Spatial Reasoning with Motion

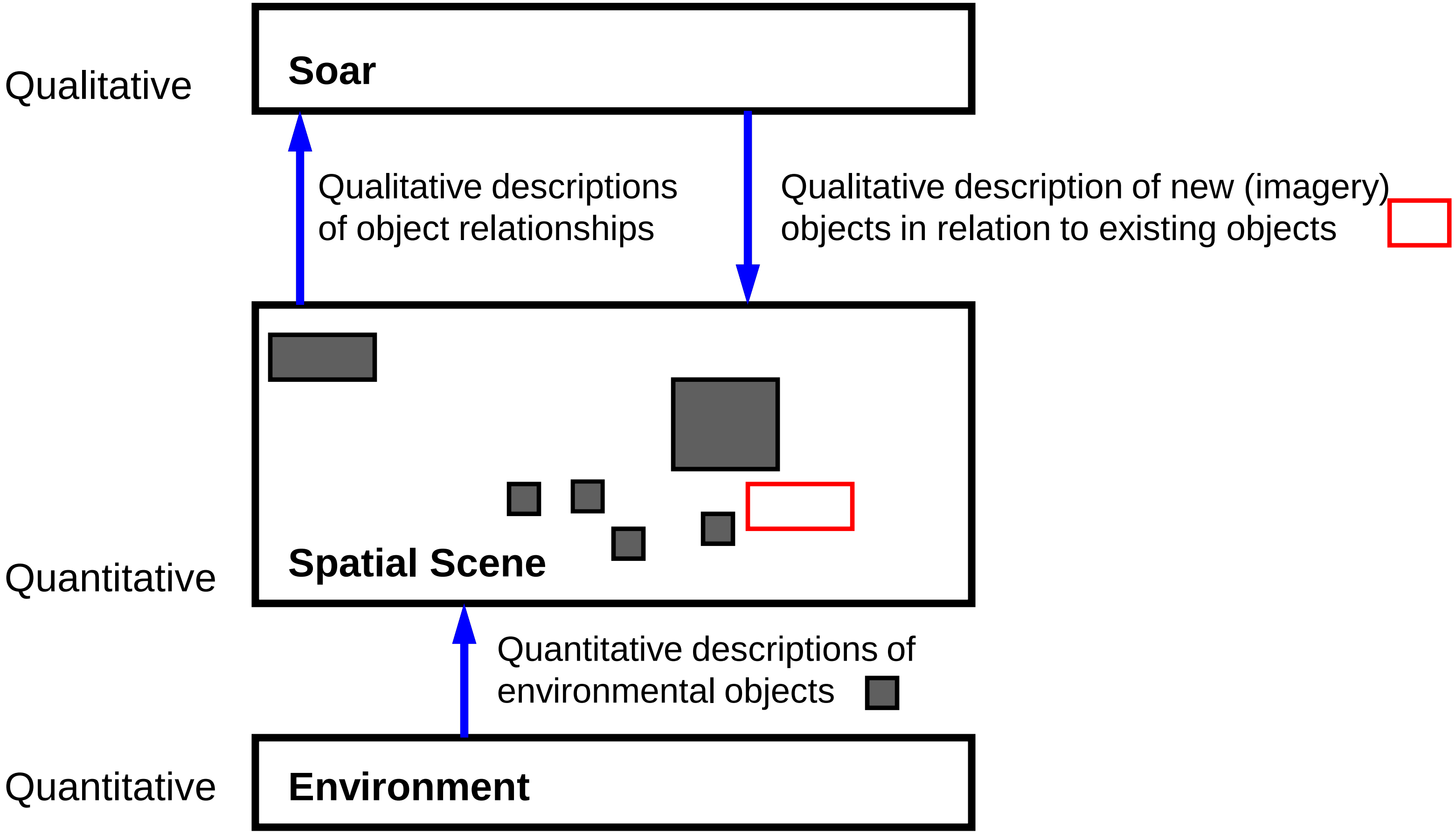
Samuel Wintermute
University of Michigan

Soar Workshop 28, Ann Arbor, MI

Overview

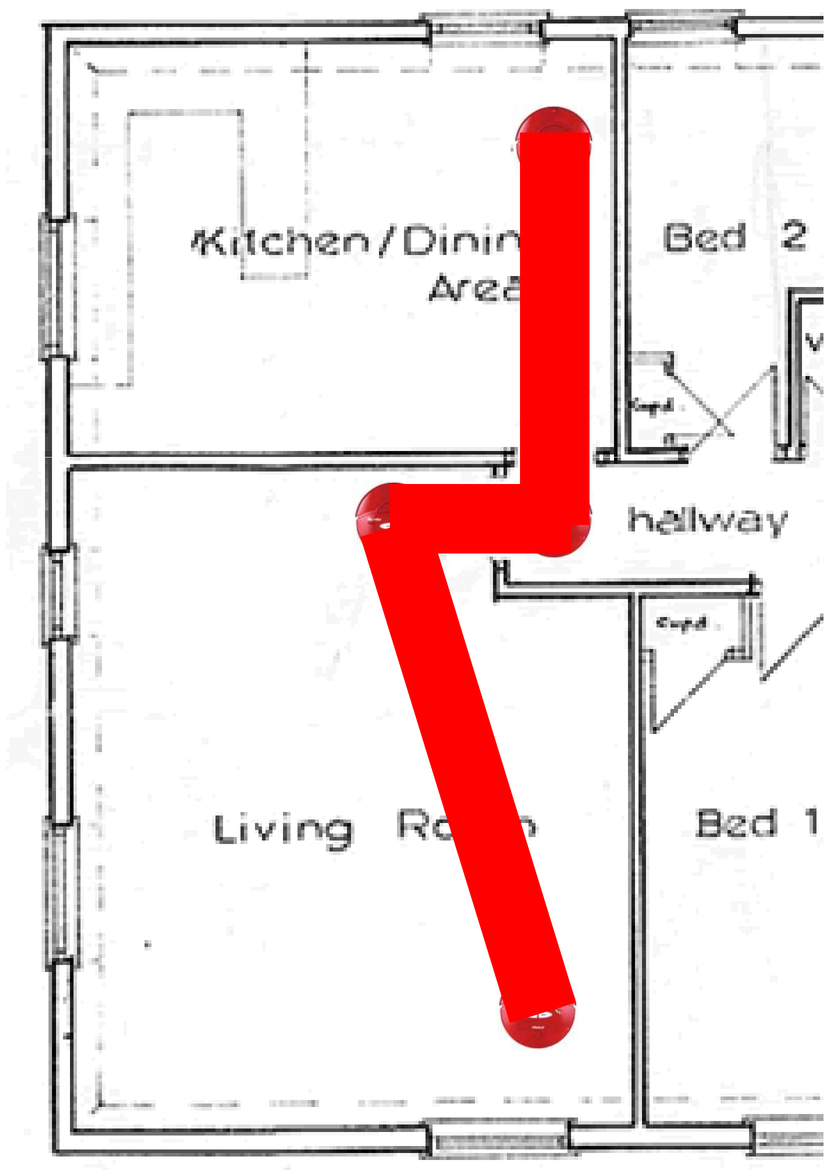
- Bimodal spatial reasoning
- Types of motion problems
- Motion models
- Examples
- Conclusion (Nuggets and Coal)

Bimodal Spatial Reasoning



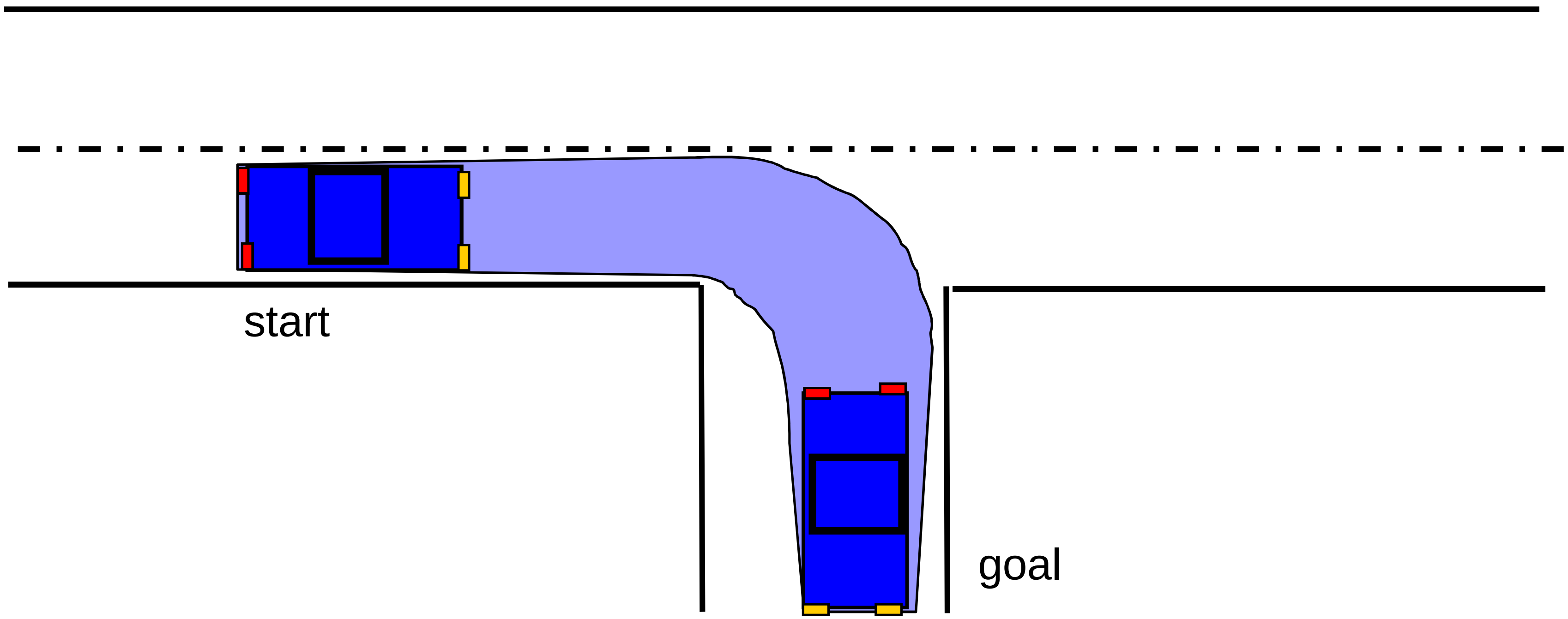
Problems with Motion: Action Planning

- An agent must be able to see the consequences of its actions
- In some cases, this is simple geometry problem



But what about more complicated situations?

- The consequence of a motion isn't always simple to represent



Other types of motion

- Not all important motion is directly related to effectors
  - Indirect actions, actions of others, environmental motion
- A system should ideally be able to reason about any type of motion it might encounter
- How can we generally represent and reason about motion?

Motion Models

- Idea: the agent should learn and replay motion patterns it perceives in the environment
  - How can these patterns be represented and controlled?

    (ignoring learning for now)

Motion Models

- Forward simulations, in the spatial level
  - Continuously transform spatial objects, based on low-level quantitative calculations

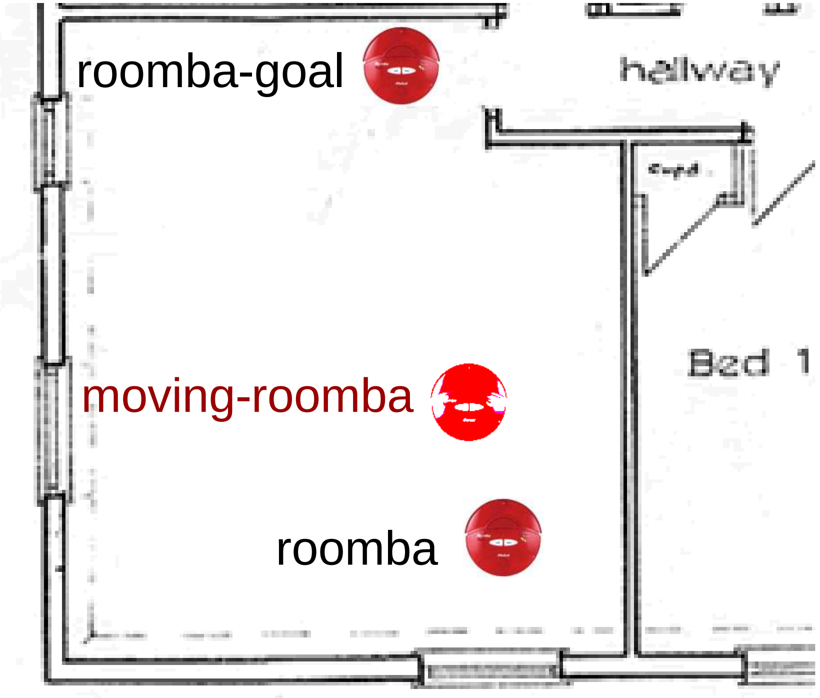
$$\dot{x} = u_s \cos\theta \qquad \dot{y} = u_s \sin\theta \qquad \dot{\theta} = \frac{u_s}{L} \tan u_\phi .$$

- Invoked and controlled by Soar
  - Soar handles qualitative aspects of problem solving
    - Soar knows object identities, e.g., what is moving, what is an obstacle
    - Soar can invoke motion as a subpart of a broader symbolic problem-solving process

Running a simulation

- Typical model interface to Soar:
  - Soar specifies:
    - a moving object
    - a goal object
    - a time step
  - Model then creates an image of the moving object after the given time step

```
^simulation
  ^type translation
  ^moving-object roomba
  ^goal-object roomba-goal
  ^time 2
```



Termination

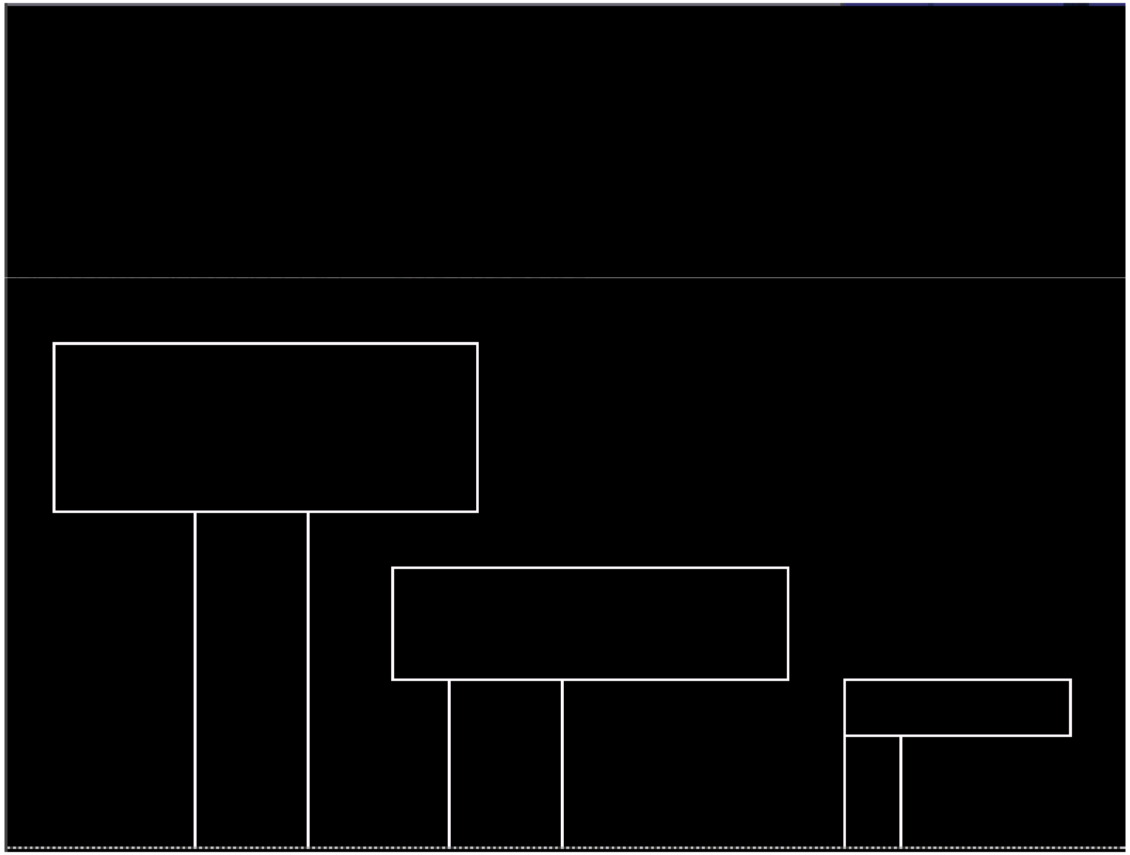
- The Soar agent is responsible for terminating simulations
- This is done by extracting predicates from the scene (e.g., "**`moving-object`** intersects **`goal-object`**")
- Soar has access to internal simulation states, to a degree determined by its step size
  - Can detect collisions here, via intersection queries
  - Speed/accuracy tradeoff

Implemented motion models

- translate: move towards or away from another object
- translate-around: move around the border of another object
- car: simple car equations, steer towards a goal object
- falling-block: simulate the effect of gravity on a block, relative to one reference block

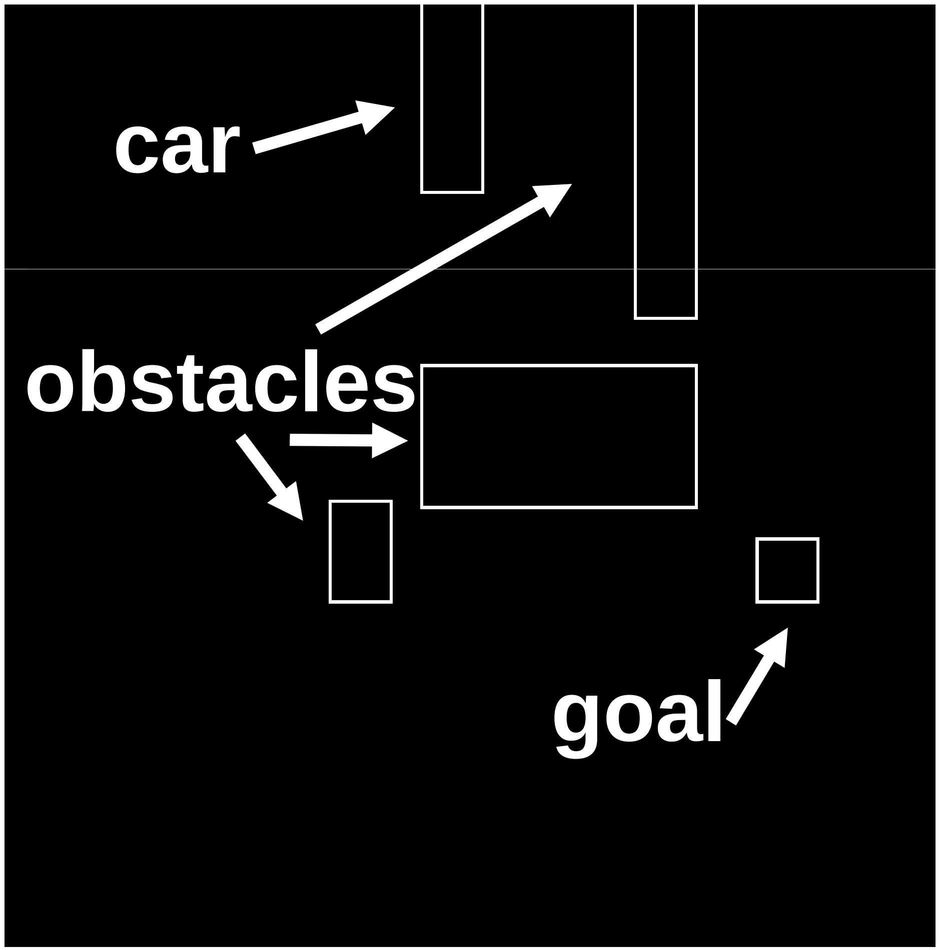
Falling block example

- Based on Funt (1980): determine which blocks will fall to the ground



Car path planning example

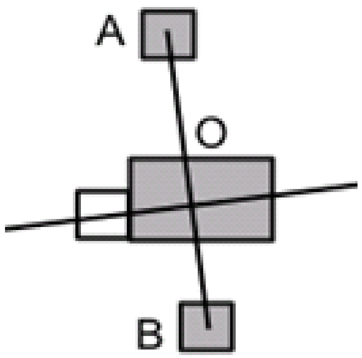
- Based on work using SRS to do qualitative path planning
- Create waypoint, try to reach waypoint, repeat



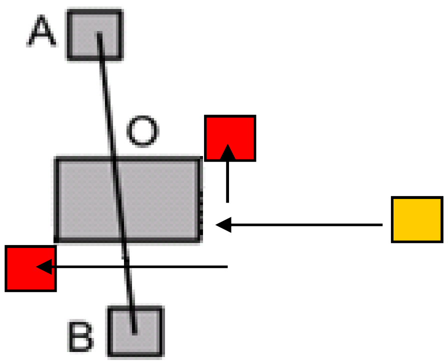
Simplifying car path planning

- Previous demonstration relied on complicated symbolic structures to describe waypoints
  - The waypoint around obstacle O, on the way from A to B is an object outside of O, on a line perpendicular to the line from A to B, and near the line from A to B
- This can be simplified using motion

with symbolic description

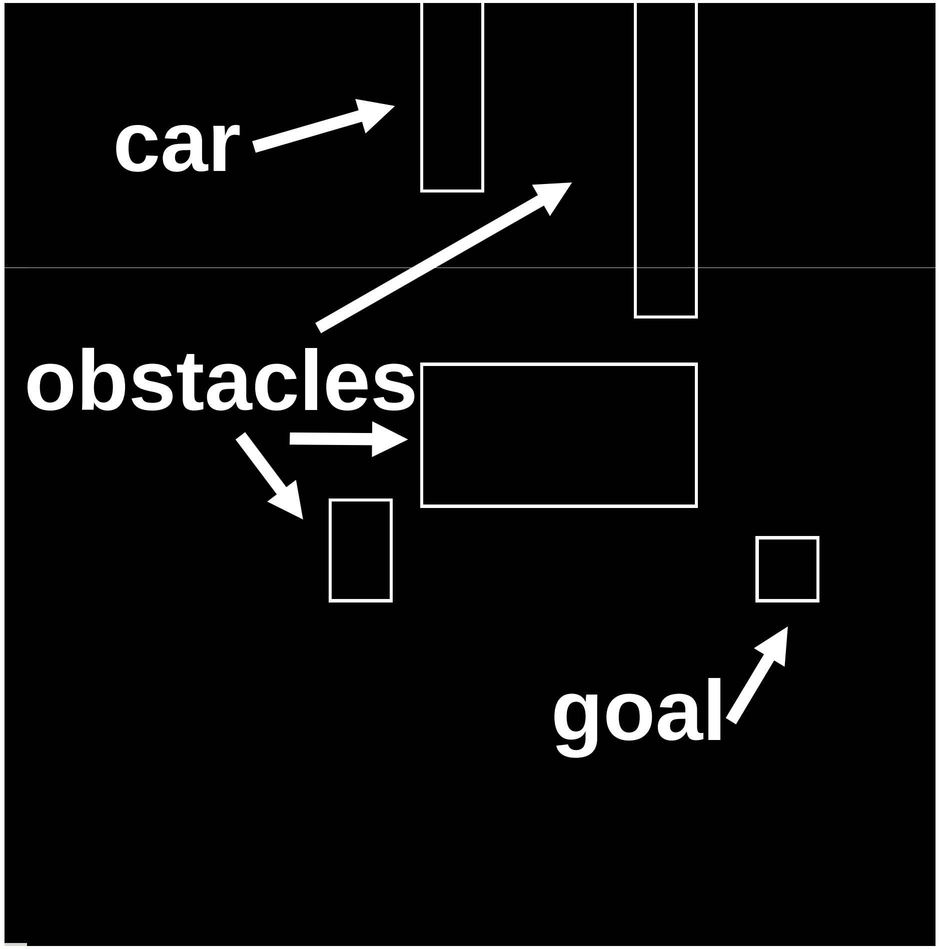


with motion



Car path planning with motion waypoints

- This approach takes longer, but is conceptually simpler
- Also, much easier to implement, since it relies on less-complex image placement capabilities



Motion Models and Action

- Motion models should be strongly connected to the action system
  - Maintaining a model can be used to help control, by speeding up the feedback cycle
  - Actual actions should be invoked and controlled by Soar in the same way imagined actions are (just as actual objects are perceived the same way as imagined objects)
- Imagery re-uses the perception system for general cognition, adding motion models allow it to re-use the action system.
- Not every motion model will have an associated action, though

Motion Model Nuggets

- Allows precise situational behavior as part of a general reasoning system
  - Soar with imagery can describe arbitrary hypothetical situations
    - “what if I was in my enemy’s position?”
    - “what if my car was a tank?”
    - …
  - Motion models can precisely interpret these situations
- Indicates a direction for grounding reasoning in reality
  - Models learned from perception of motion can be used in non-motion problems (e.g., placing an imaginary waypoint by sliding it around)
- Shows how to decompose action control to symbolic and sub-symbolic parts
- Can represent motion without a learning theory

Coal

- Lacks a good learning theory
  - We know it should learn from perception, and roughly how complicated the models should be, but not much more.
  - Also lacks a common representation, other than plain C++.
- Completeness issues
  - The system sometimes must arbitrarily choose (among equally-valid alternatives) where to place an object
    - This might result in missing correct solutions