15-887
Planning, Execution and Learning

Application: Examples of Planning for Mobile Manipulation and Articulated Robots

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Two Examples

- Planning for Mobile Manipulation
- Planning for Articulated Robots

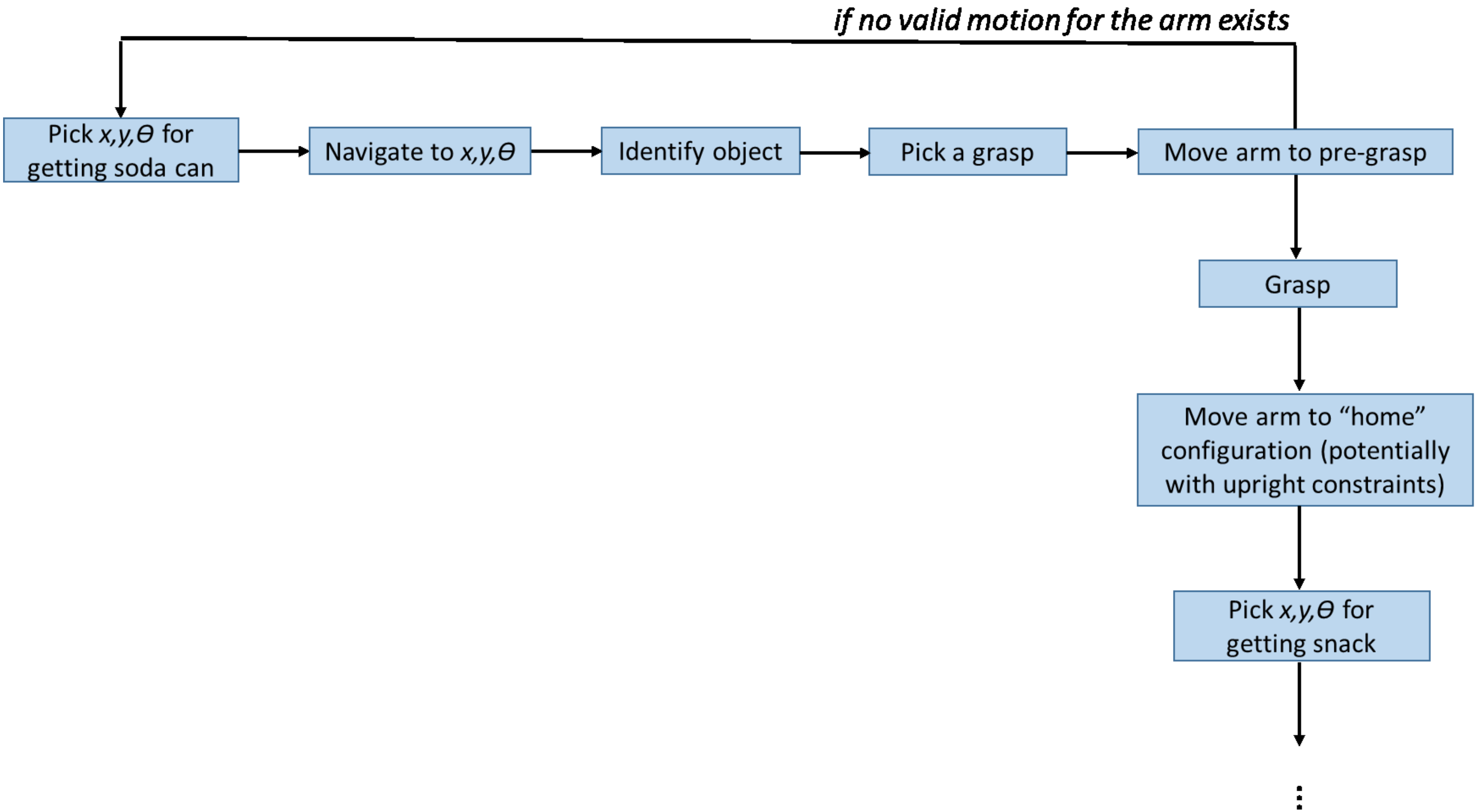
Two Examples

- Planning for Mobile Manipulation
- Planning for Articulated Robots

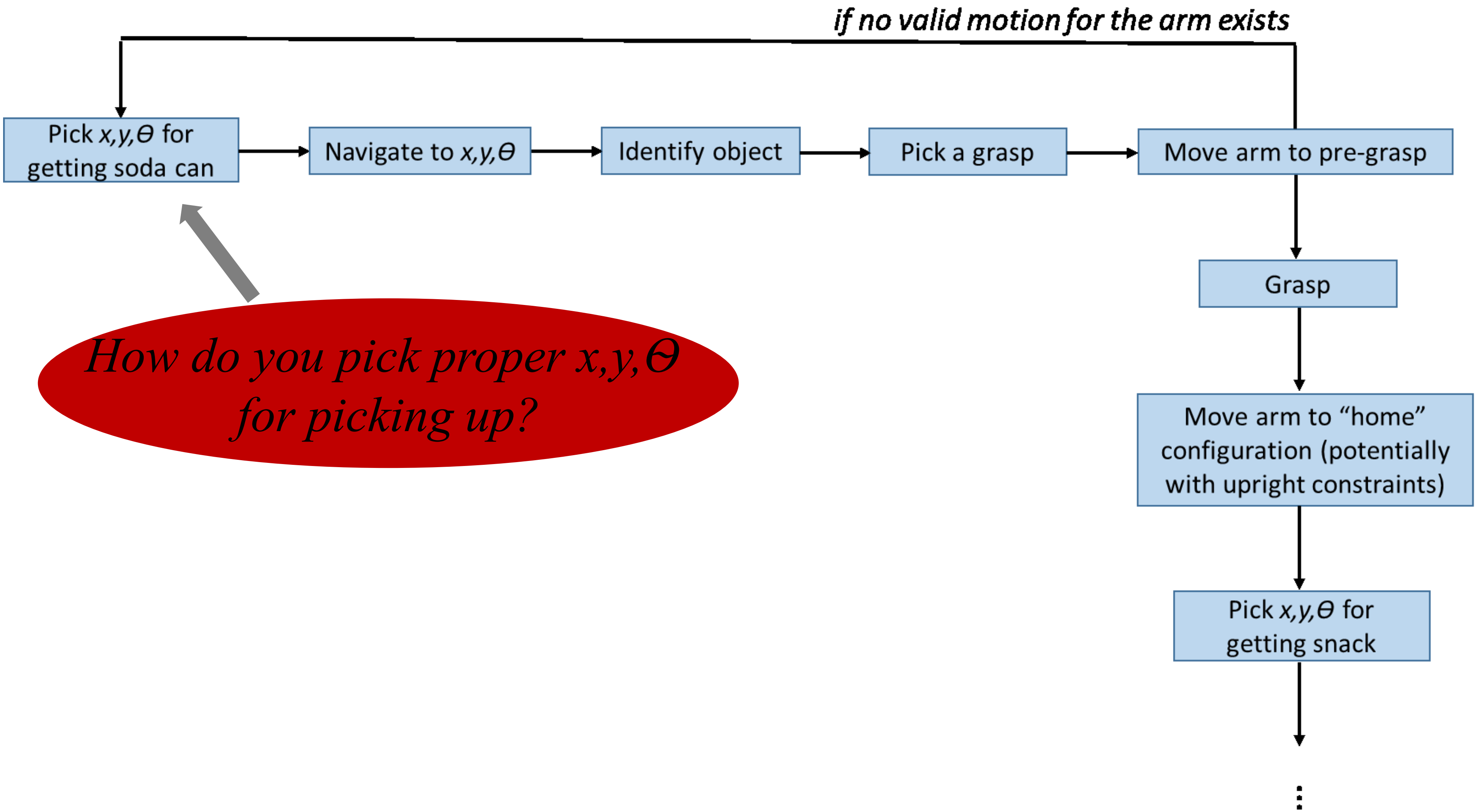
Robotic Bartender Demo ([Phillips et al.])

- Robot takes in a command from User Interface as to what soda can and snack to deliver

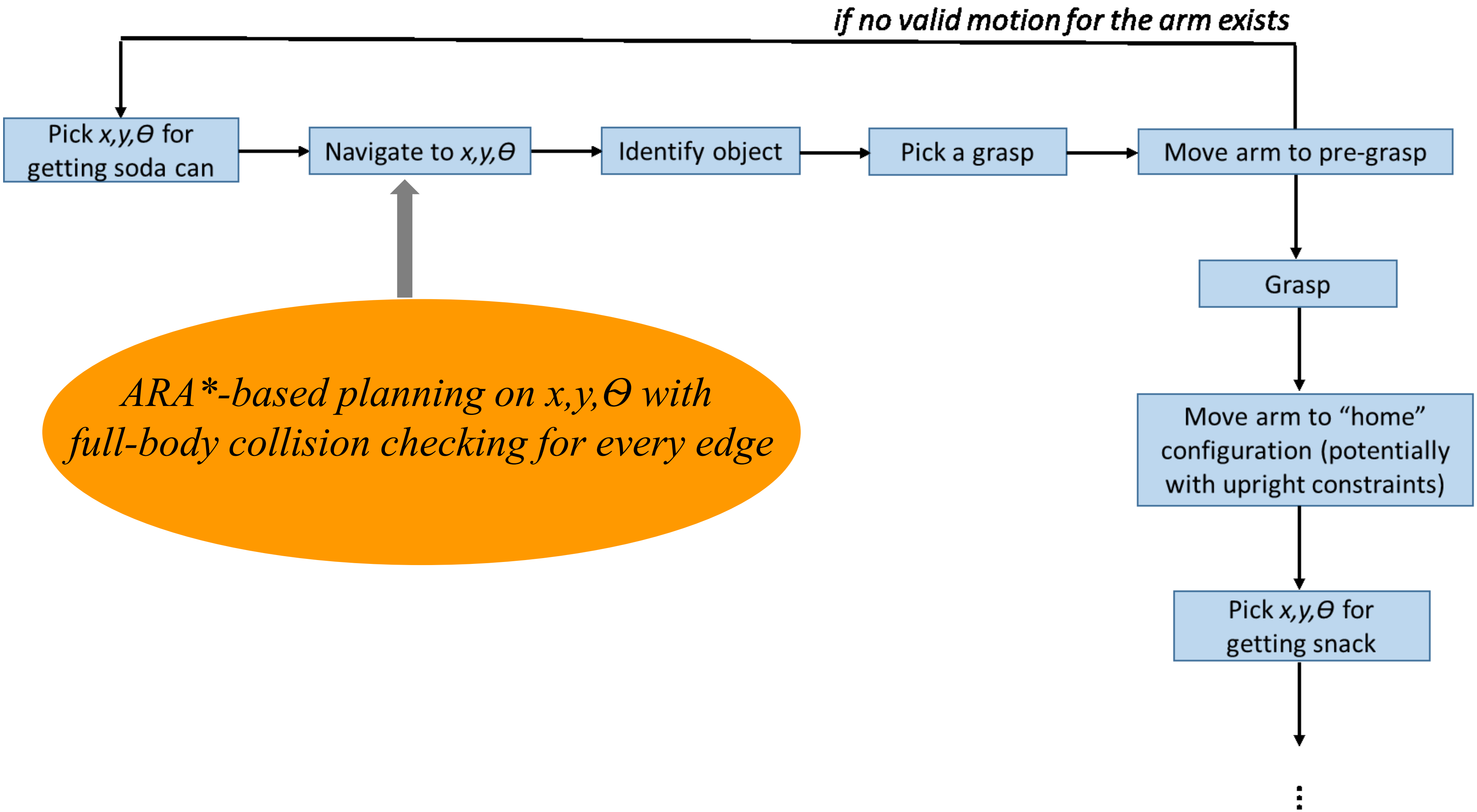
Typical Sequence of Operations (State Machine)



Typical Sequence of Operations (State Machine)

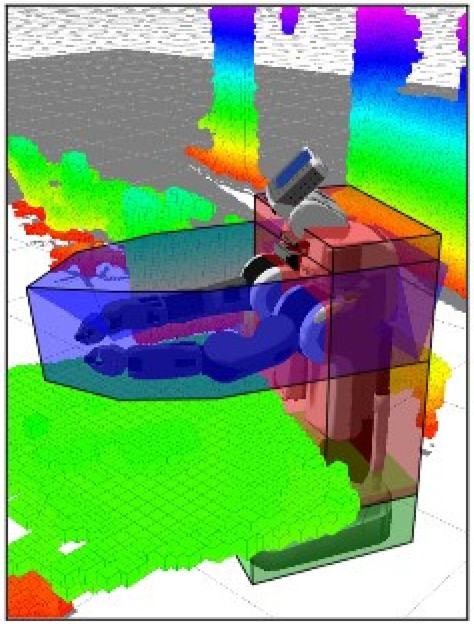


Typical Sequence of Operations (State Machine)



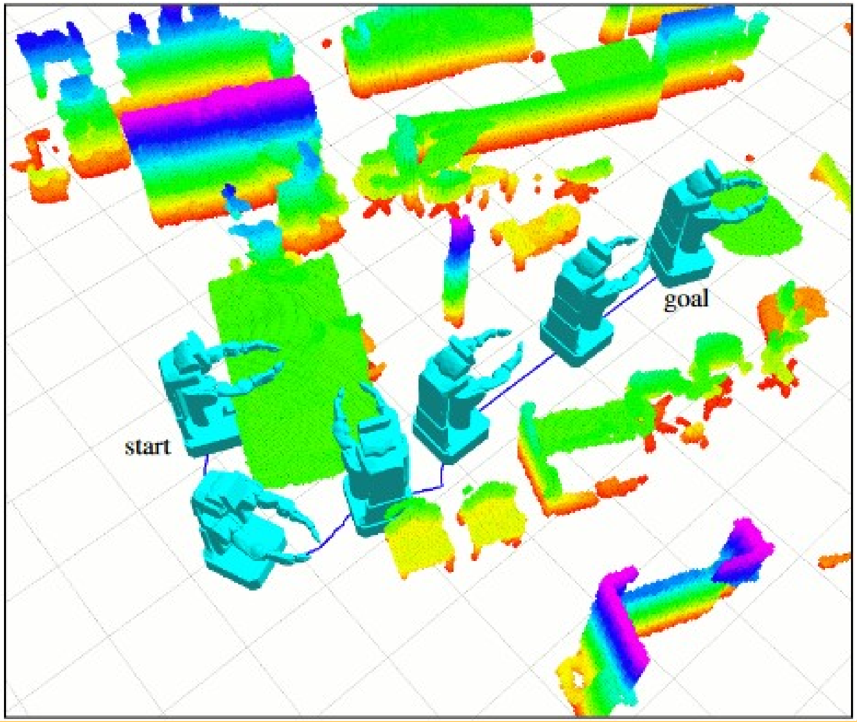
Graph for Navigation with Complex 3D Body [Hornung et al., '12]

- 3D (x,y,θ) lattice-based graph representation for full-body collision checking
  - takes set of motion primitives as input
  - takes N footprints of the robot defined as polygons as input
  - each footprint corresponds to the projection of a part of the body onto x,y plane
  - collision checking/cost computation is done for each footprint at the corresponding projection of the 3D map

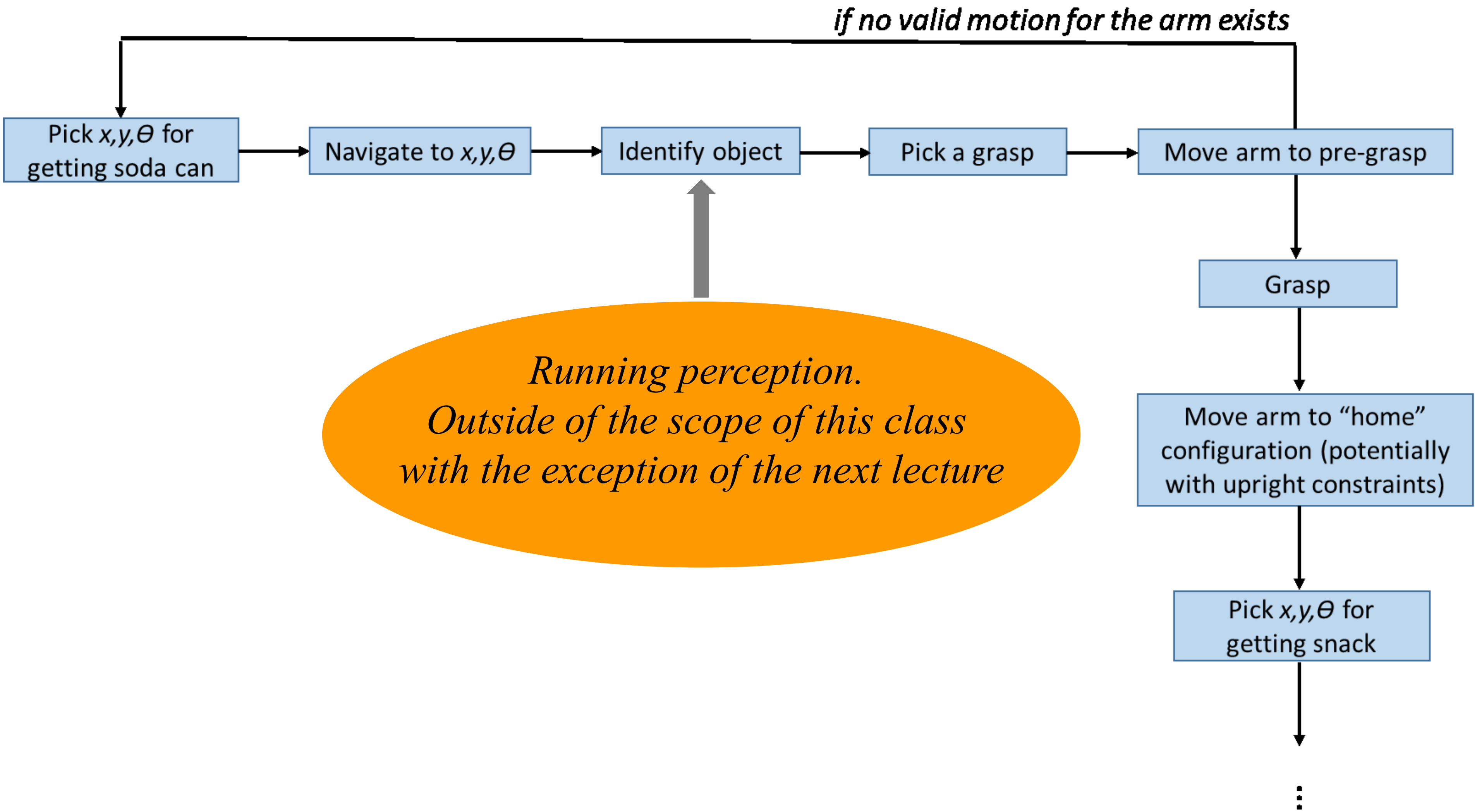


Graph for Navigation with Complex 3D Body [Hornung et al., '12]

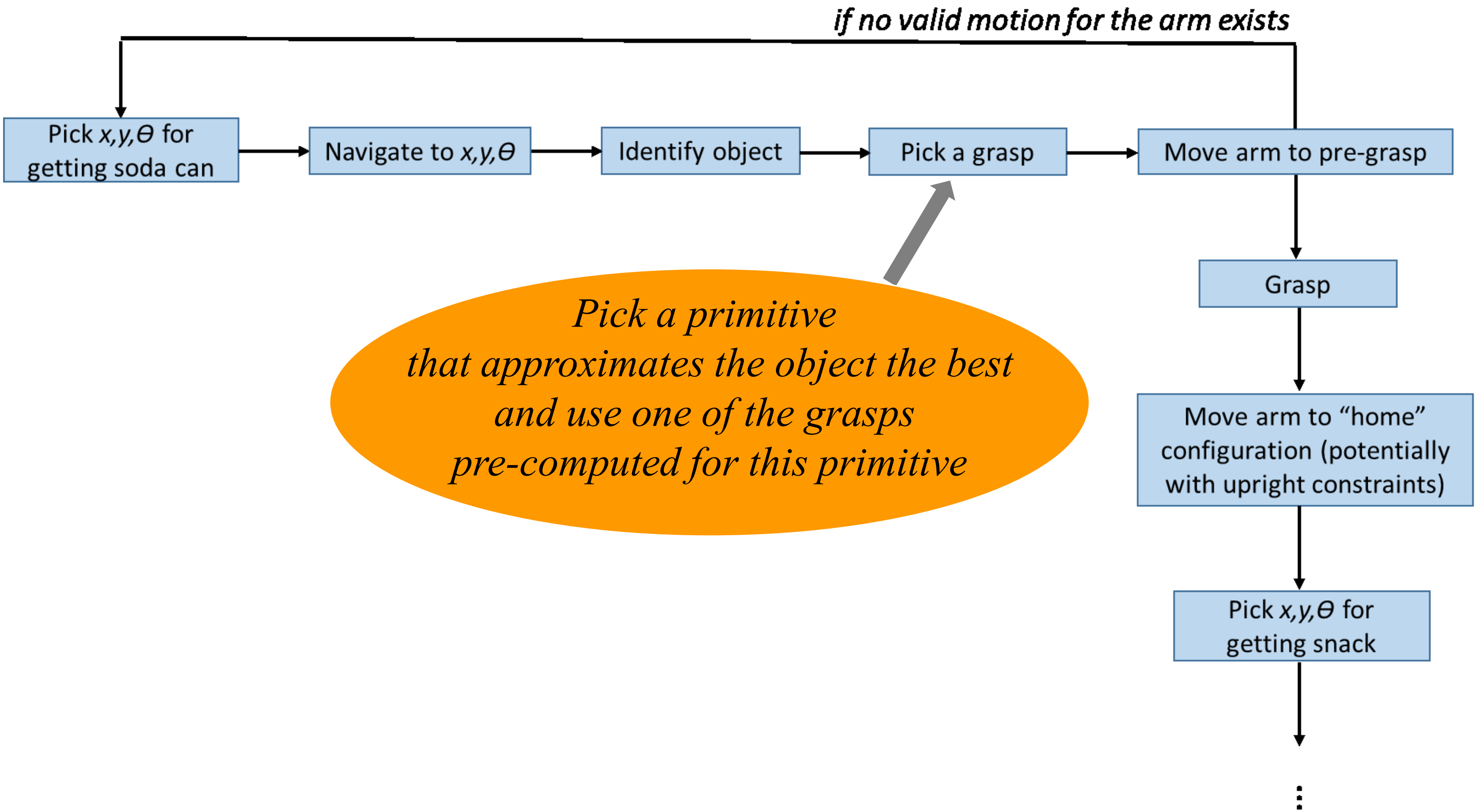
- 3D (x,y,θ) lattice-based graph representation for full-body collision checking
  - takes set of motion primitives as input
  - takes N footprints of the robot defined as polygons as input
  - each footprint corresponds to the projection of a part of the body onto x,y plane
  - collision checking/cost computation is done for each footprint at the corresponding projection of the 3D map



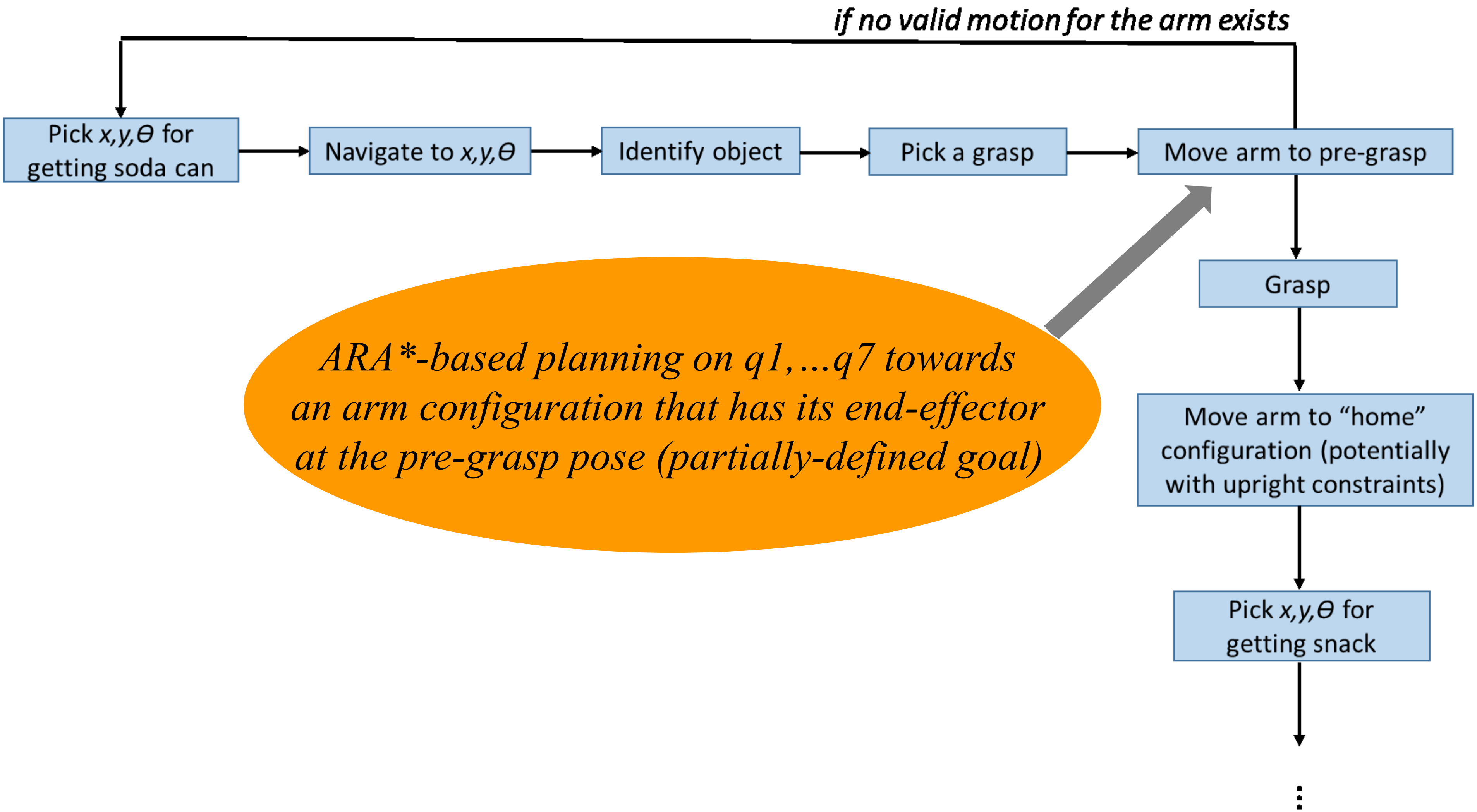
Typical Sequence of Operations (State Machine)



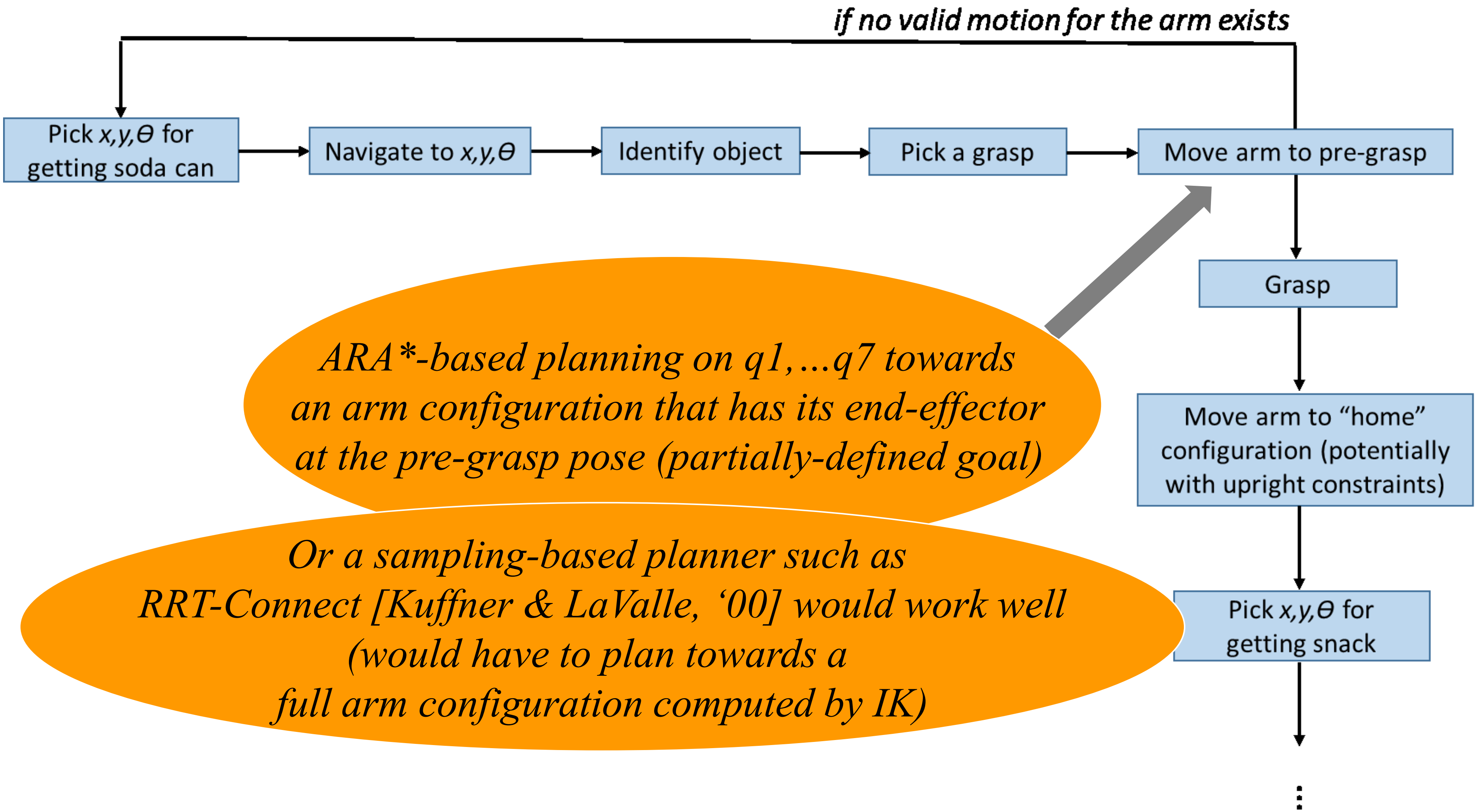
Typical Sequence of Operations (State Machine)



Typical Sequence of Operations (State Machine)



Typical Sequence of Operations (State Machine)

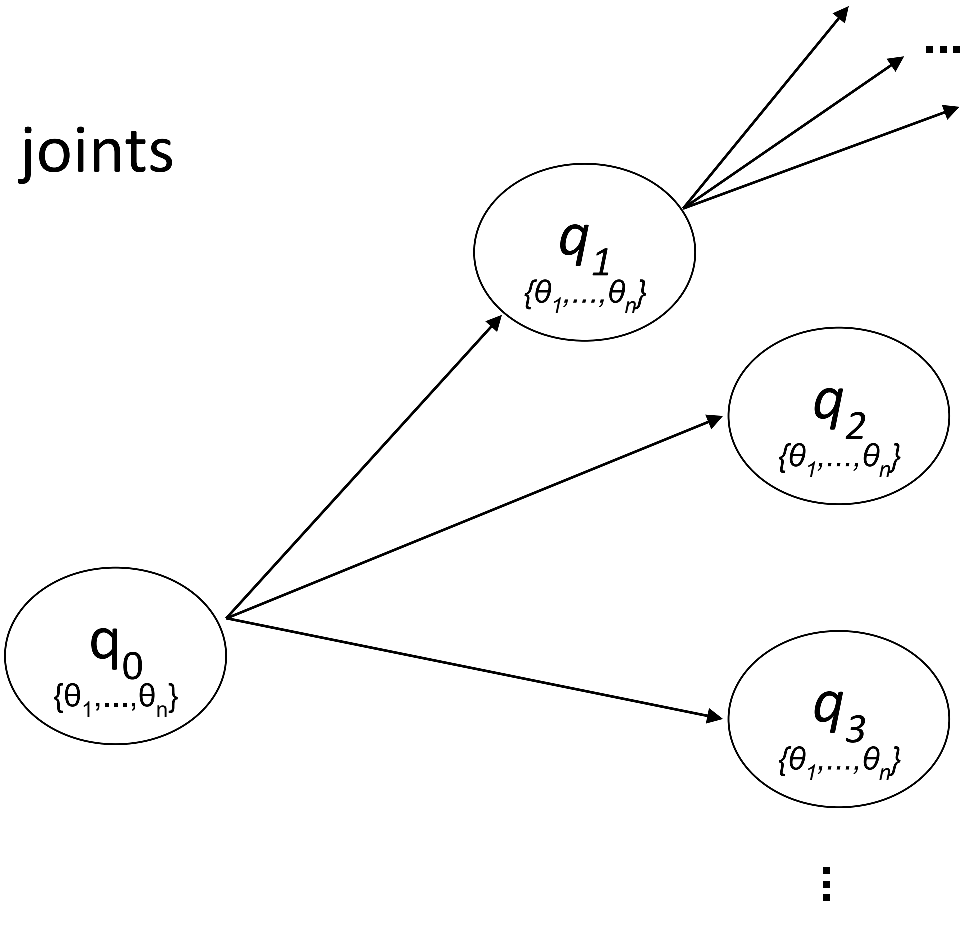


ARA-based planning on q1,…q7 towards an arm configuration that has its end-effector at the pre-grasp pose (partially-defined goal)*

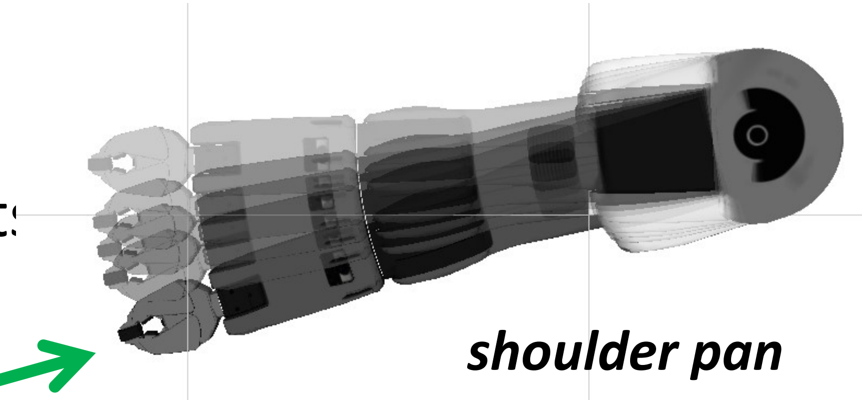
Or a sampling-based planner such as RRT-Connect [Kuffner & LaValle, '00] would work well (would have to plan towards a full arm configuration computed by IK)

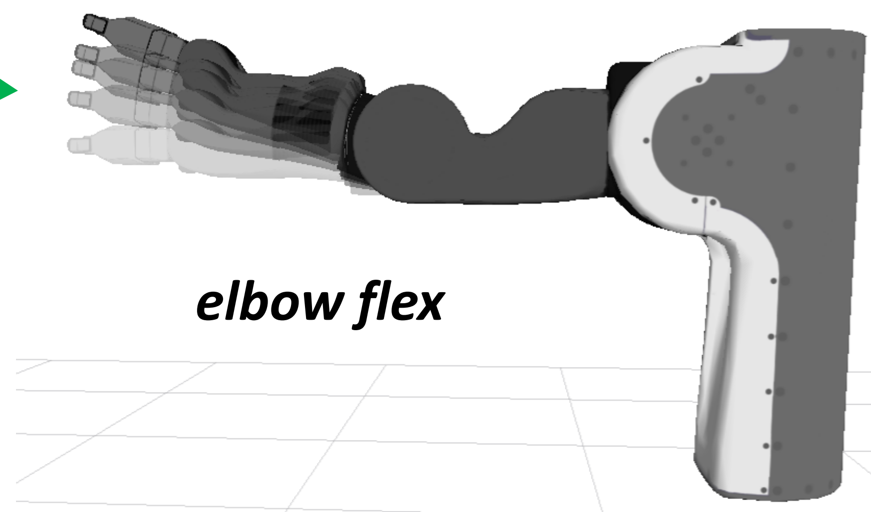
Manipulation Lattice for Arm Planning [Cohen et al., '13]

- Representation
  - ex. Single arm with n joints
    - $\{\theta_1,\ldots,\theta_n\}$



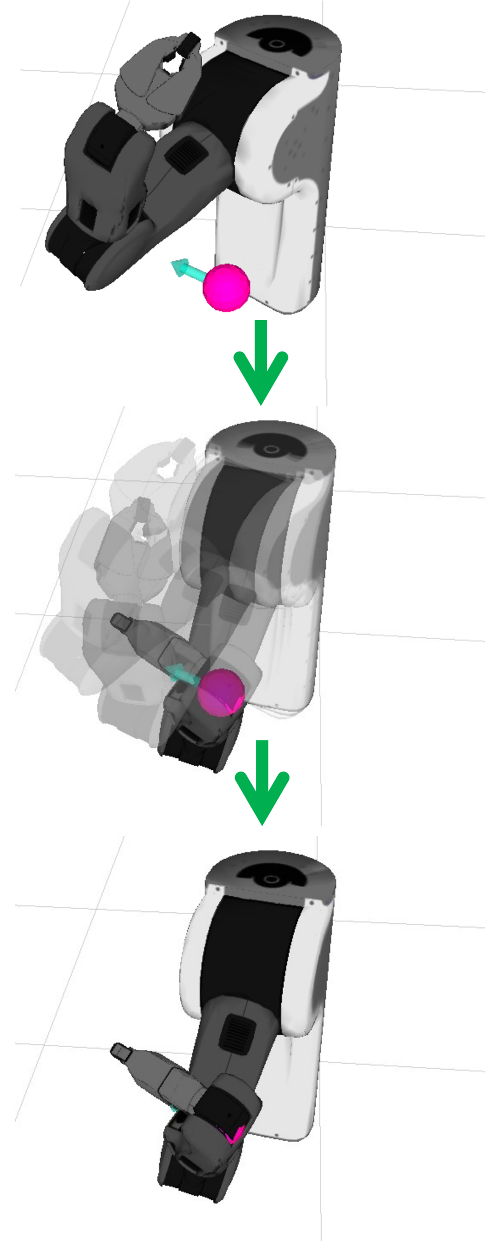
- Representation
  - ex. Single arm with n joints
    - $\{\theta_1,\ldots,\theta_n\}$
- Motion Primitives
  - **Static**





Manipulation Lattice for Arm Planning [Cohen et al., ‘13]

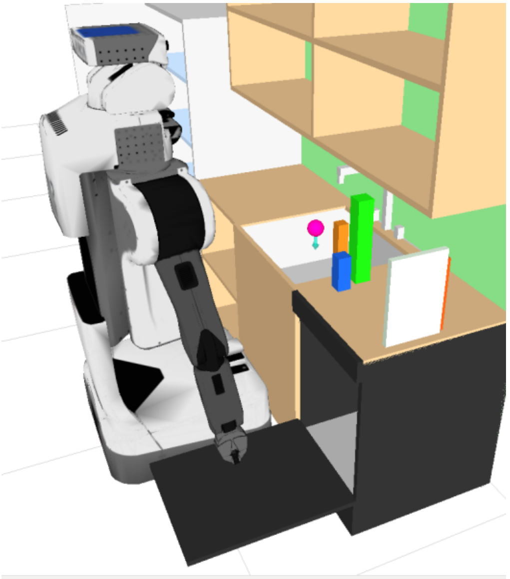
- Representation
  - ex. Single arm with n joints
    - $\{\theta_1, \ldots, \theta_n\}$
- Motion Primitives
  - Static
  - **Adaptive**
    - Snap-to-goal motion
      - Analytical solver & IK-based (only if feasible)
    - Capable of satisfying arbitrary goal constraint despite discretization

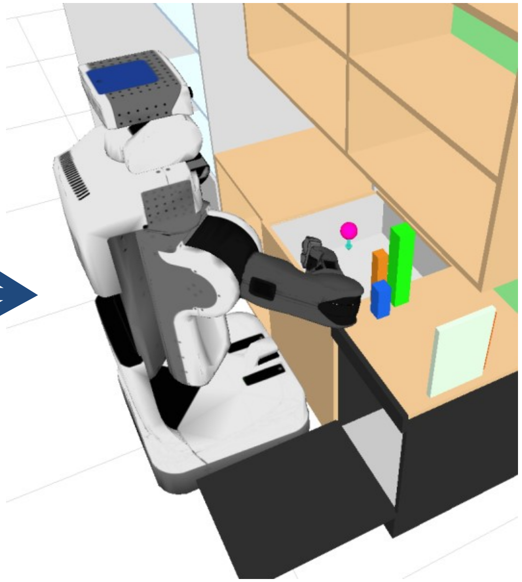


Manipulation Lattice for Arm Planning [Cohen et al., '13]

- Non-uniform Dimensionality
  - far from goal: only 4 DoFs active
  - around goal: all 7 DoFs are active
- Non-uniform Resolution
  - far from goal: larger discretization of joint angles
  - around goal: finer discretization of joint angles

4D
1. Shoulder pan
2. Shoulder pitch
3. Upper arm roll
4. Elbow flex





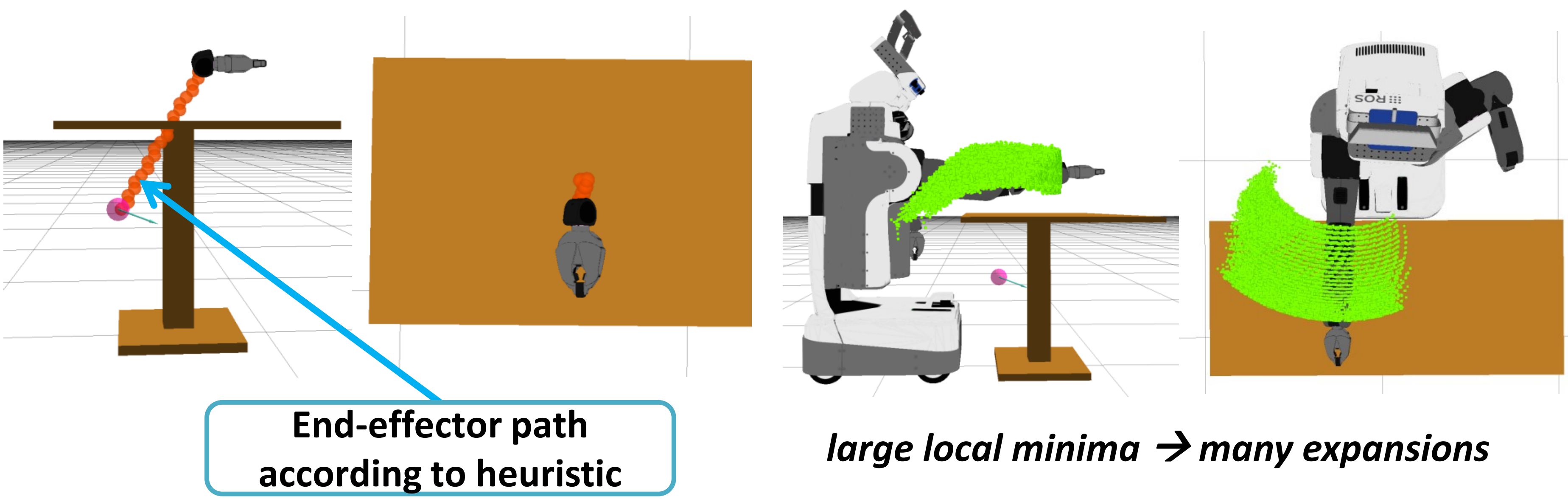
7D
1. Shoulder pan
2. Shoulder pitch
3. Upper arm roll
4. Elbow flex
5. Forearm roll
6. Wrist flex
7. Wrist roll

Informative Heuristics for Arm Planning [Cohen et al., '13]

Any ideas?

- Euclidean distance for the end-effector?
  - **deep local minima!**

$h_{\text{Euclidean}}$

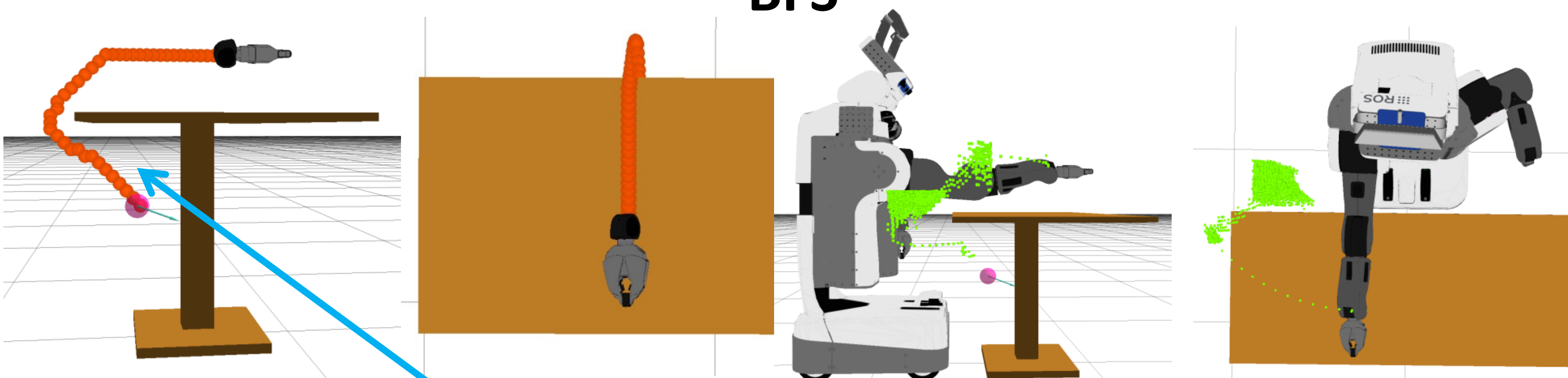


large local minima → many expansions

Informative Heuristics for Arm Planning [Cohen et al., '13]

- 3D BFS for end-effector accounting for obstacles
  - computes distances from each <x,y,z> to the goal for the end-effector sphere
  - **dramatically better than Euclidean distance**
  - admissible
  - fast to compute

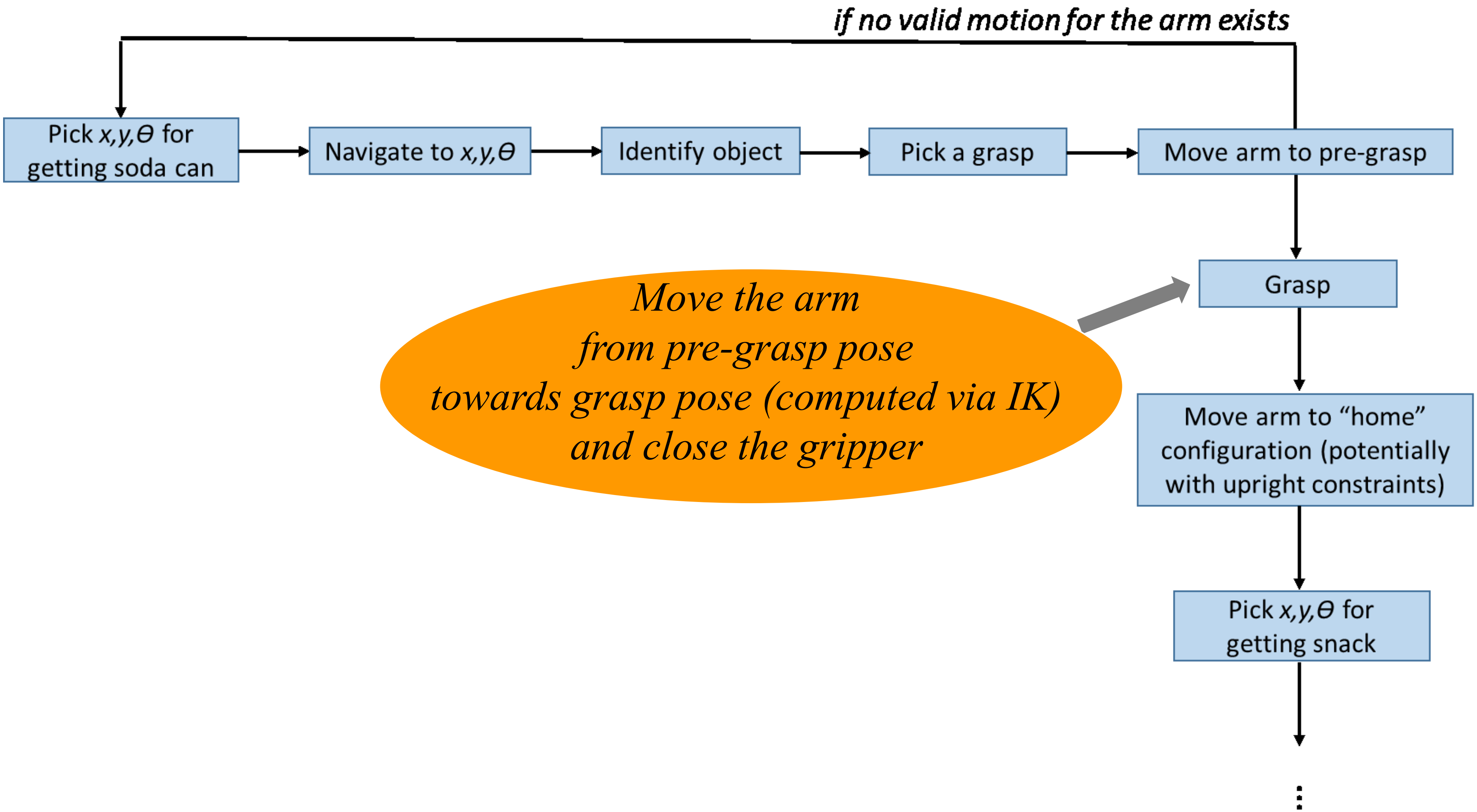
h_{BFS}



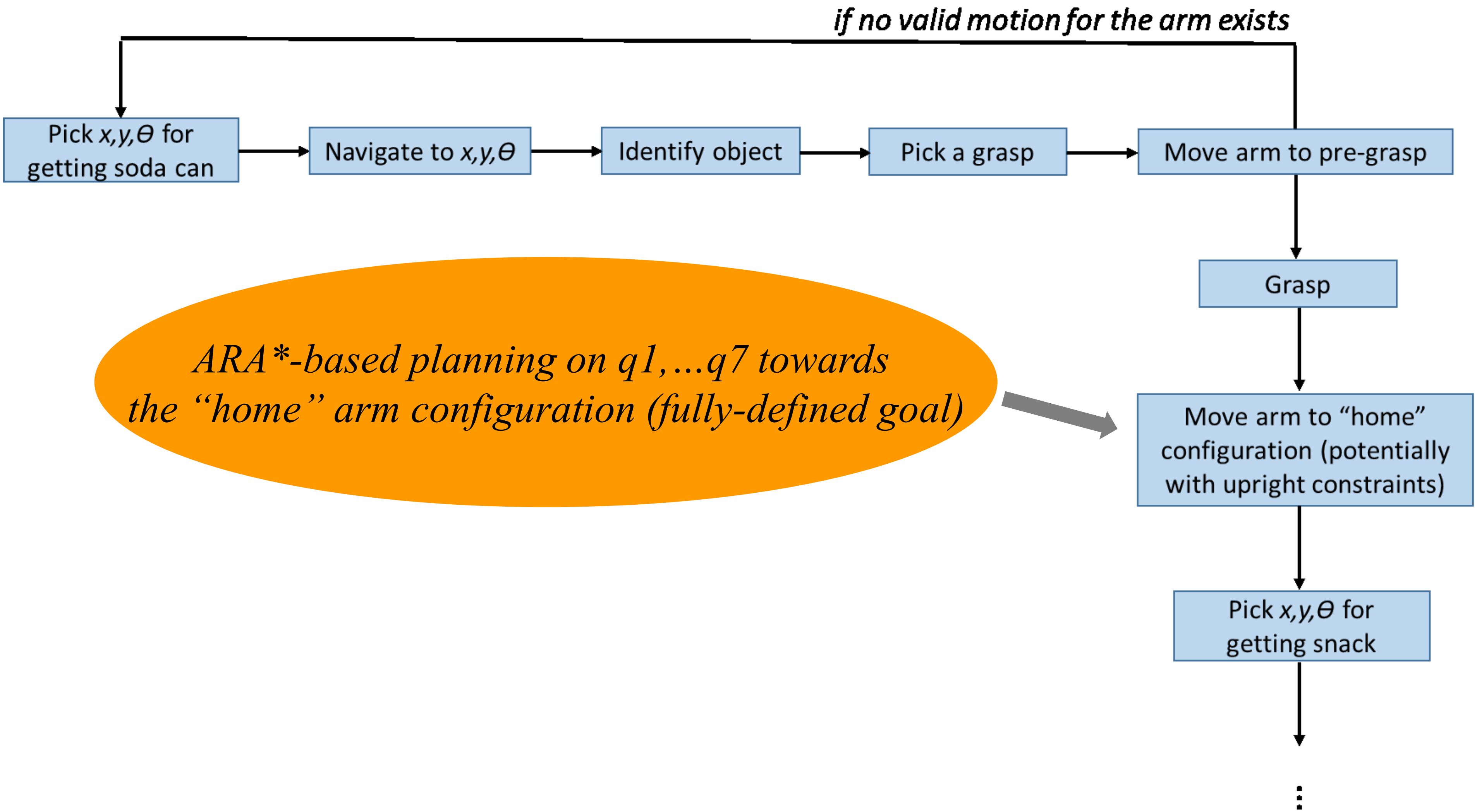
End-effector path according to heuristic

shallow local minima → fewer expansions

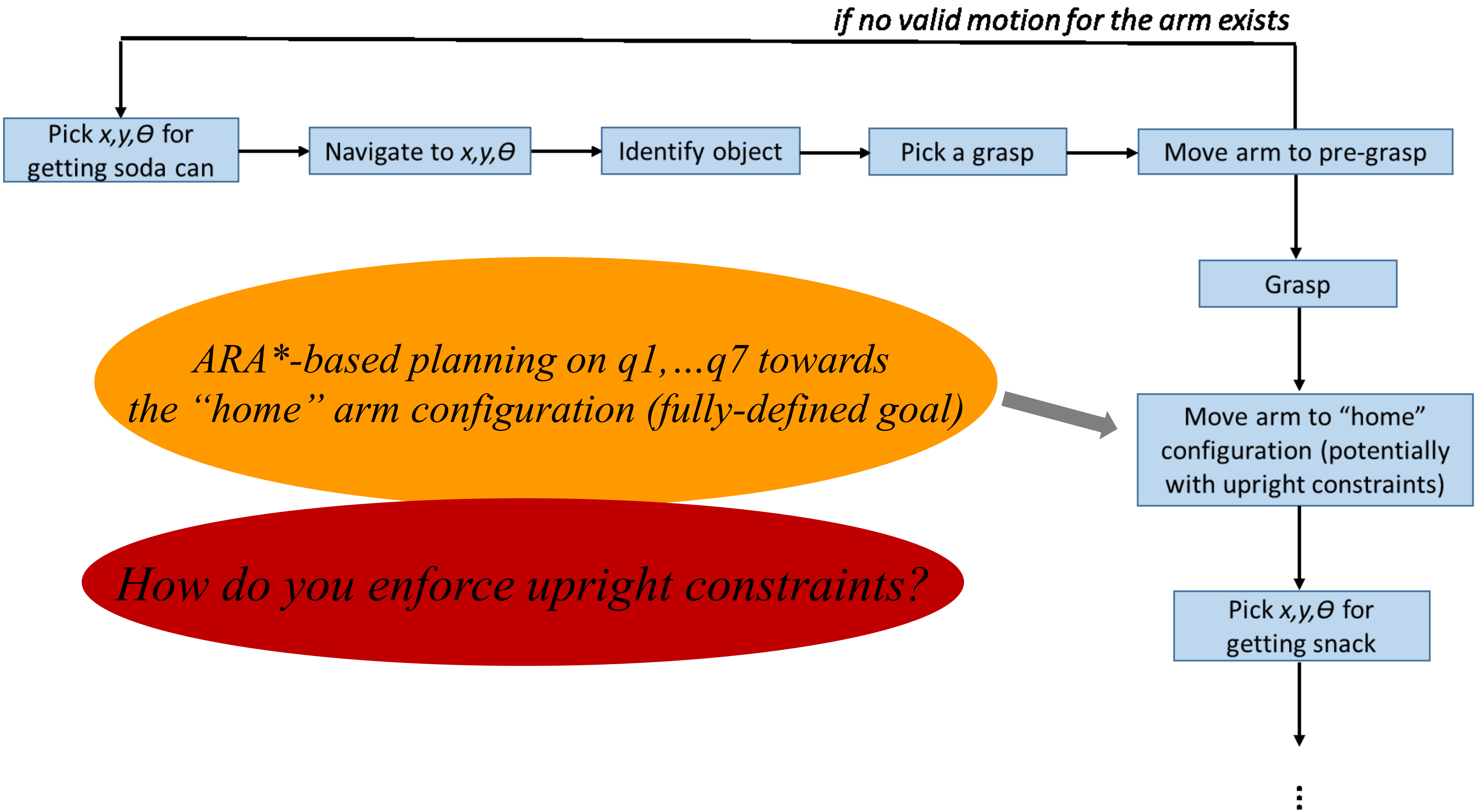
Typical Sequence of Operations (State Machine)



Typical Sequence of Operations (State Machine)



Typical Sequence of Operations (State Machine)

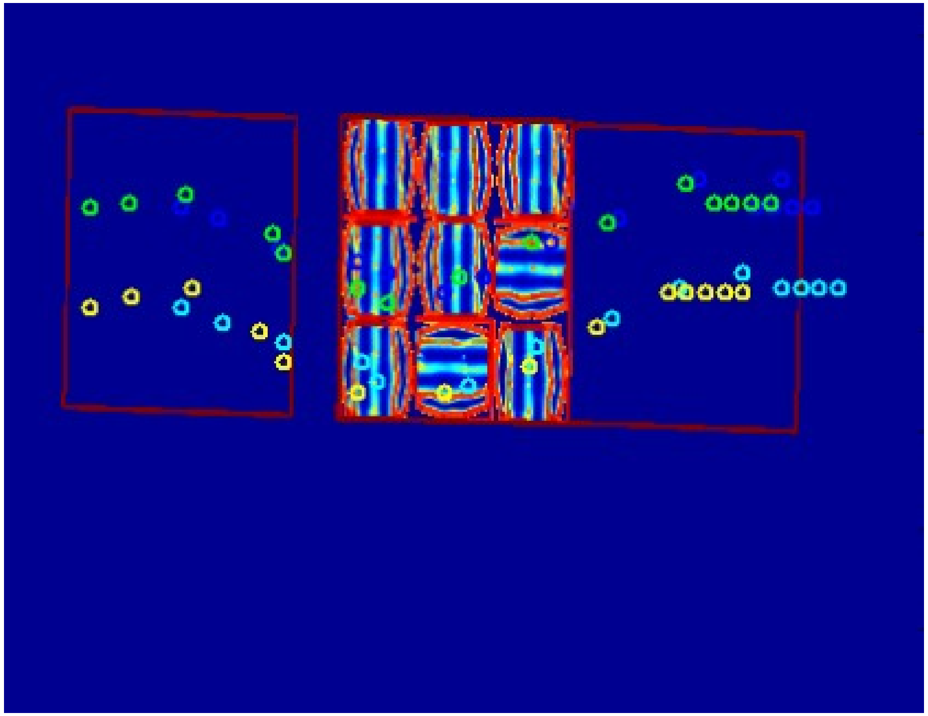


Two Examples

- Planning for Mobile Manipulation
- Planning for Articulated Robots

Little Dog Demo [Vernaza et al., ‘09]

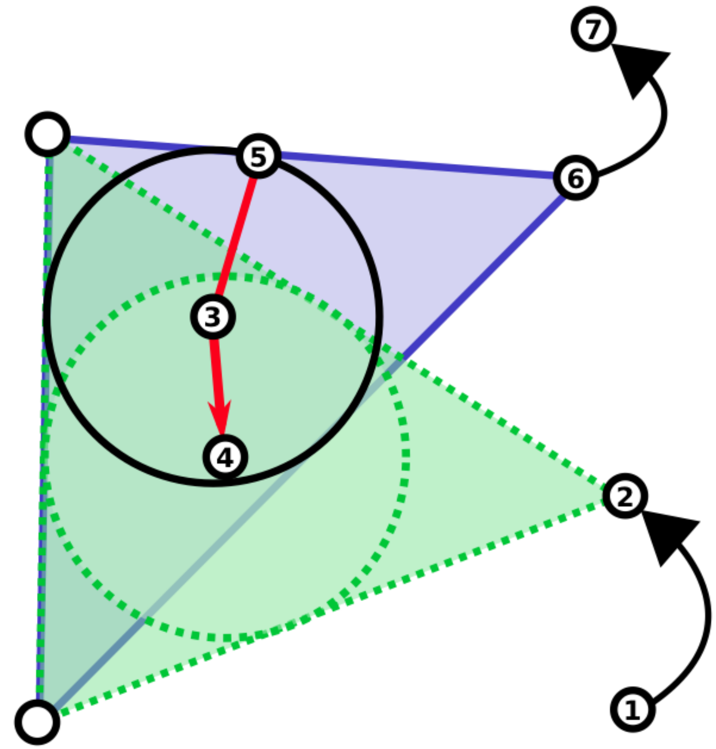
- Little Dog robot needs to traverse a fully-known terrain
- Planning
  - Plans footsteps first with an anytime variant of A*
  - Compute COM of the robot afterwards to support execution



Footstep Planner [Vernaza et al., ‘09]

Assumptions of the planner:

- Only one leg lifted at a time to ensure static stability
- Center of mass shifts during quad-support phase to prevent tipping
- Footholds chosen deliberately to maximize stability



Footstep Planner [Vernaza et al., ‘09]

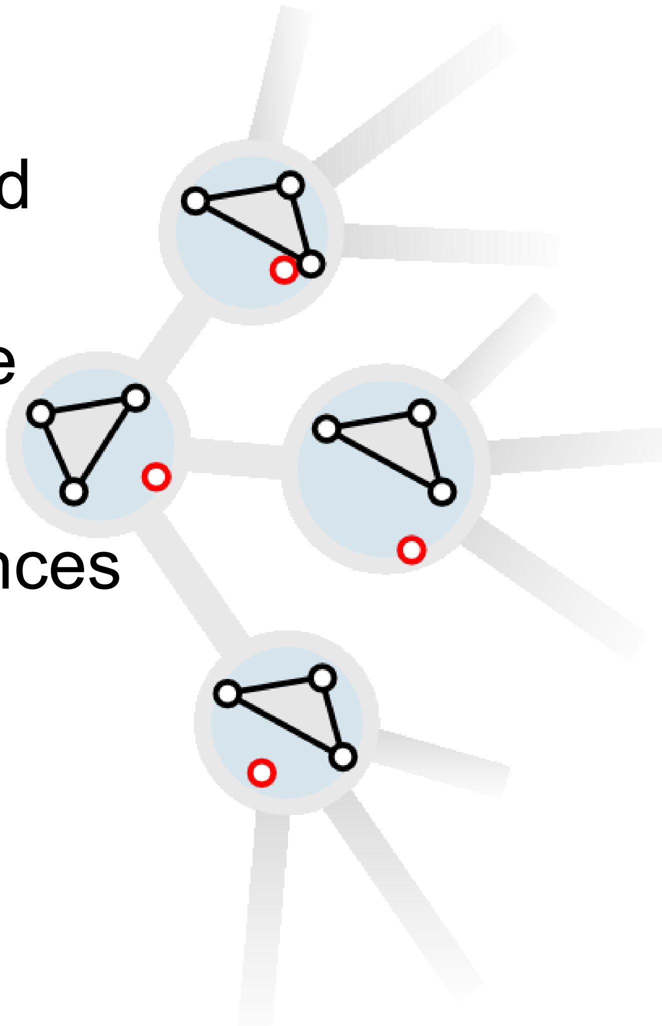
Planner builds (on-the-fly/implicit) Stance Graph:

- **Node (stance):** 9-dimensional foothold configuration

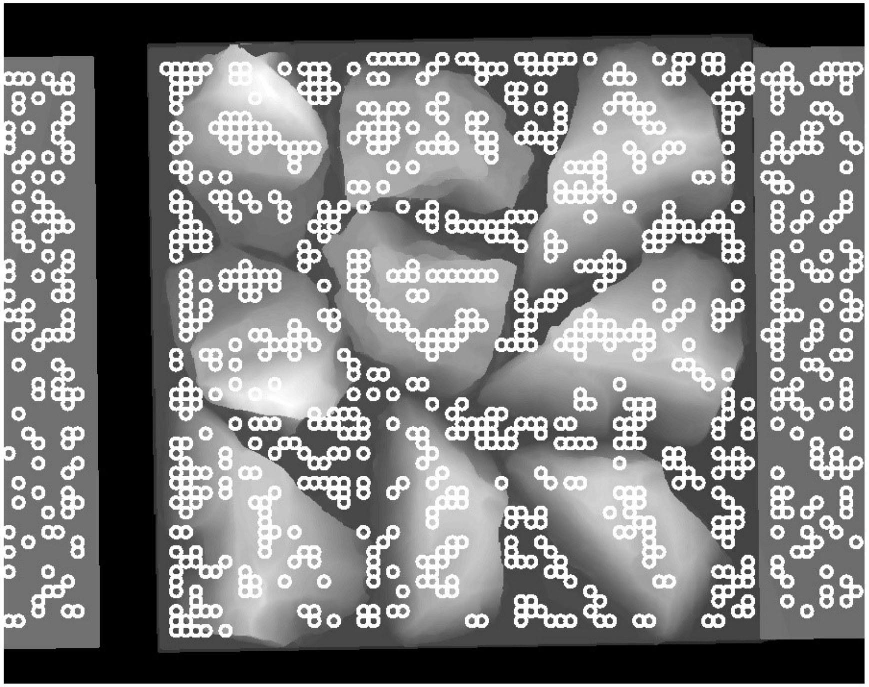
  - feet positions and current gait phase

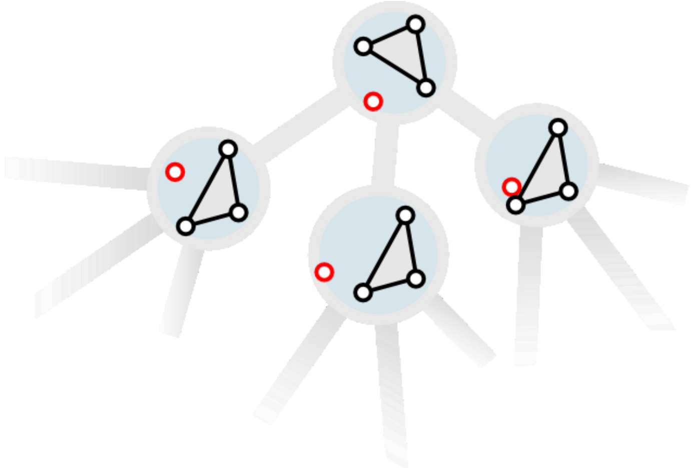
- **Edge**: feasible transition between stances

- **Edge costs** for transitions computed based on risk, anticipated delay



Implementation of *GetSuccessors(s)* Function

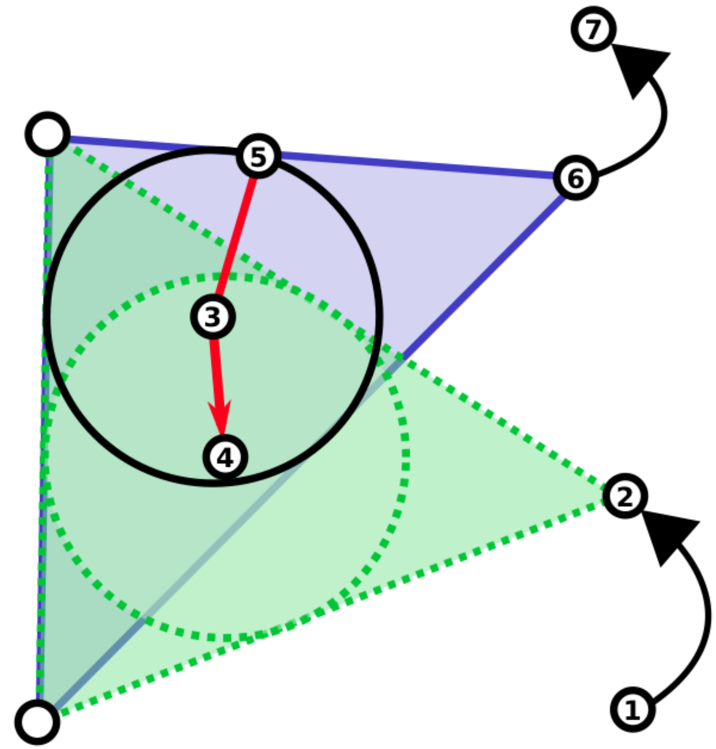




- Finite set of good quality candidate footholds selected prior to planning
- Valid stances are kinematically feasible 4-tuples of candidate footholds
- Successors of a given stance computed by:
  - determining reachable candidate footholds that result in a valid stance

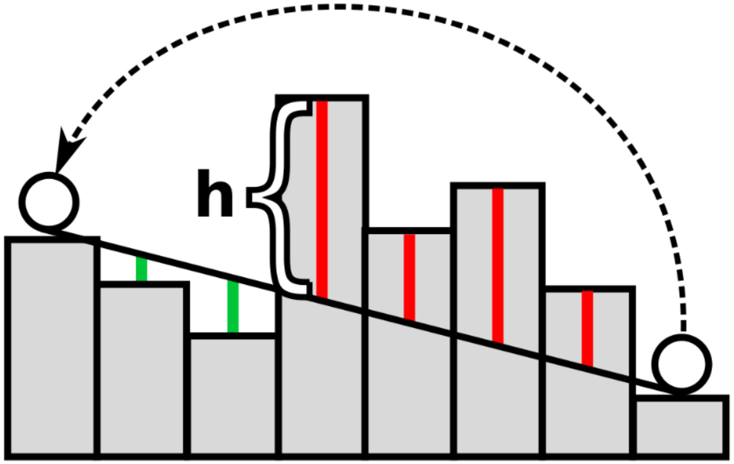
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Overhead**
    - Fixed cost per step
  - **Center of mass travel**
    - Discourages backwards motion of COM
  - **Incircle radius**
    - Farthest distance from interior to exterior of support triangle



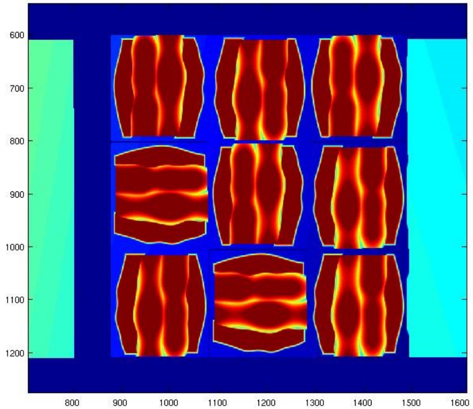
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Collision**
    - Risk of body/foot colliding with terrain
  - **Foot height variance**
    - Encourages robot to stay level



Implementation of *GetCost(s,s')* Function

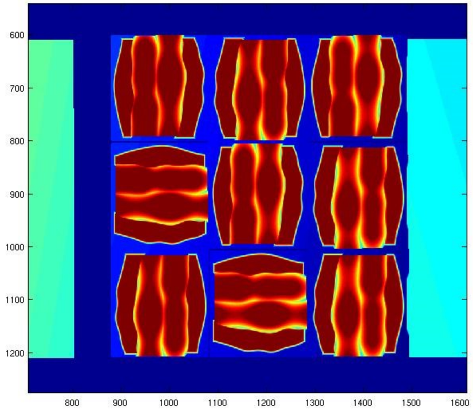
- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to reach next foothold, switch to next support triangle without dragging feet
  - **Terrain slope**
    - Ensures terrain slope supports direction of motion
  - **Terrain cost**
    - Considers slippage potential given terrain



Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to r… support triangle without d…
  - **Terrain slope**
    - Ensures terrain slope supports direction of motion
  - **Terrain cost**
    - Considers slippage potential given terrain

Lots of features make up the cost function.
Fine tuning them is not fun ☹



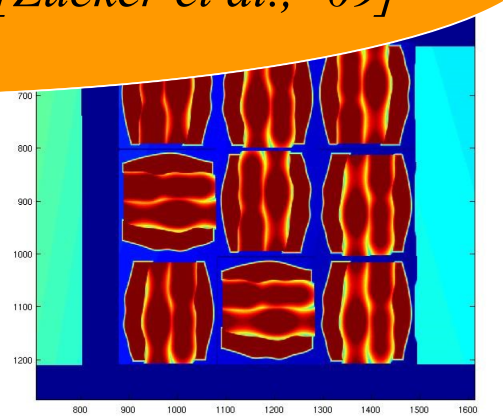
Implementation of *GetCost(s,s')* Function

- Edgecosts are weighted sum of:
  - **Reachability**
    - Robot's ability to
      support triangle without
  - **Terrain slope**
    - Ensures terrain
      directio
  - **Terrain cost**
    - Considers slippage potential given terrain

Lots of features make up the cost function.
Fine tuning them is not fun ☹

Any thoughts on making the process easier?

Using Learning Cost (Inverse RL) function we learned before!
See it being used for LittleDog by [Zucker et al., '09]



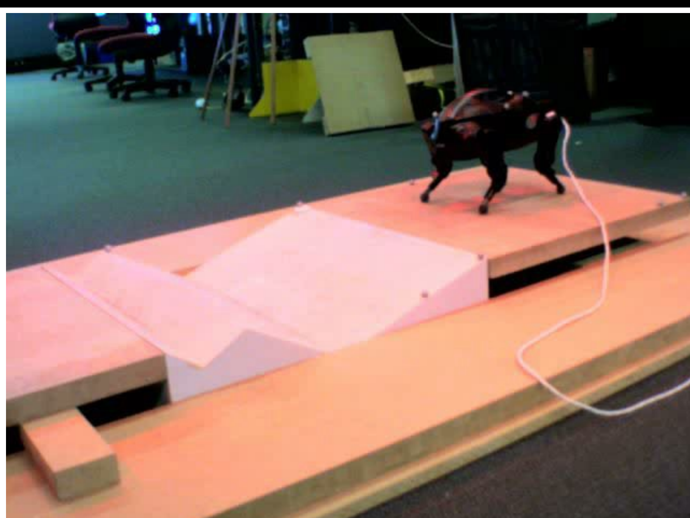
Sometimes smart but often stupid

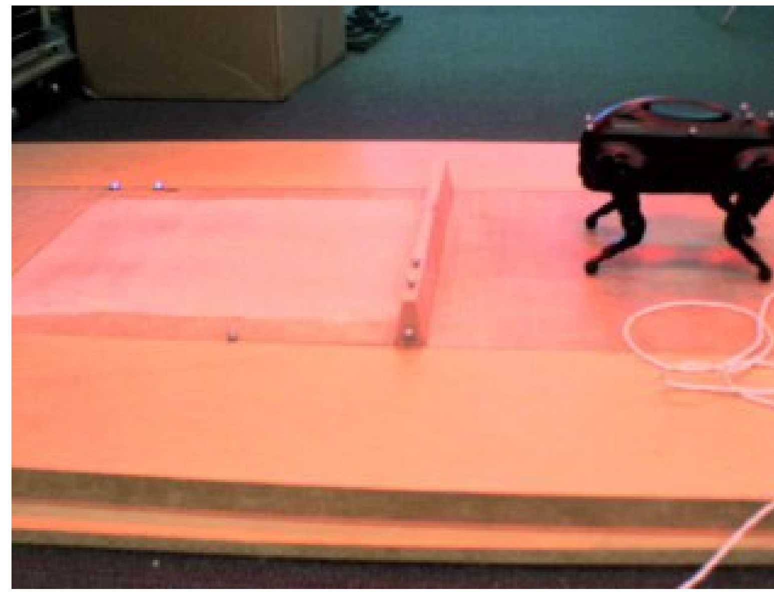
Search-based planning for a legged robot over rough terrain

Paul Vernaza, Maxim Likhachev,
Subhrajit Bhattacharya, Sachin Chitta*,
Aleksandr Kushleyev, Daniel D. Lee

GRASP Laboratory
University of Pennsylvania

*Willow Garage, Inc.





no footstep planning

Summary

- Multiple planners used for both domains
- Start and goal configurations are often most constrained
  - can be exploited by the planners
- Cost is really complex in planning for articulated robots (e.g., quadrupeds and humanoids)
- Planning is higher-dimensional but can take longer than on ground and aerial vehicles
- Design of proper heuristics is a key to efficiency