Lecture 35

CS 6260 Automated Planning and Learning

Integrated Task and Motion Planning

Department of Computer Science and Engineering

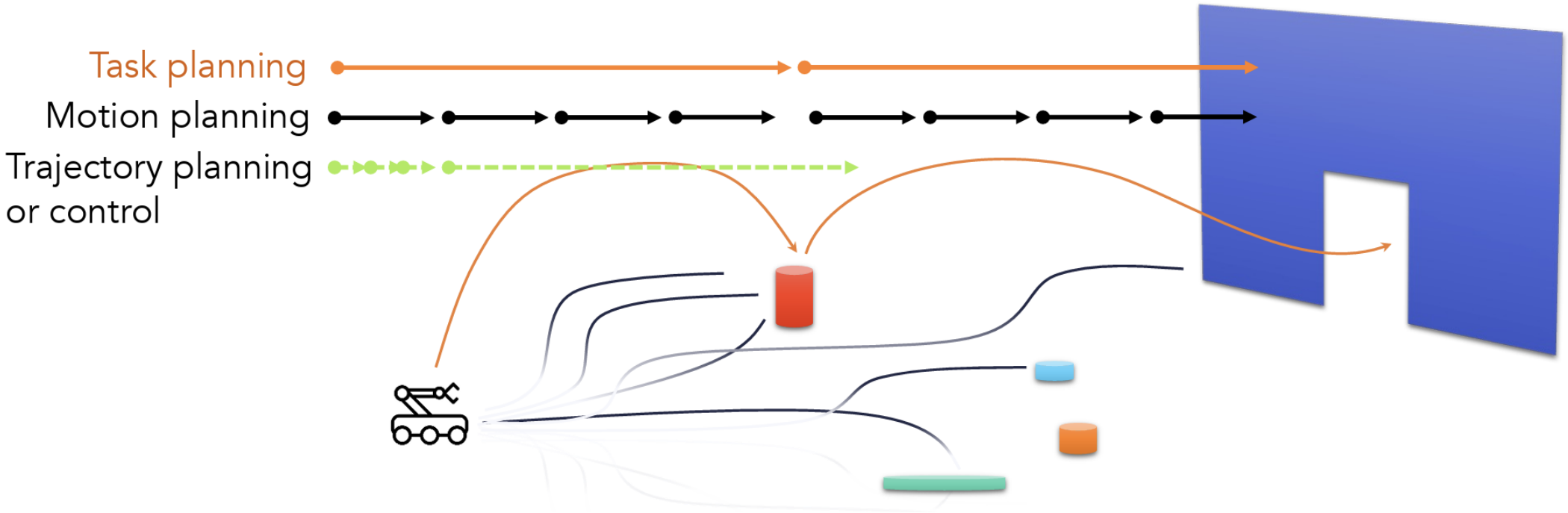
Indian Institute of Technology Madras

Material based on IJCAI 2023 Tutorial on Integrated Task and Motion Planning by Naman Shah, Sarah Keren, and Siddharth Srivastava.

We Need to Integrate Task and Motion Planning!

- Task planning – given a task planning problem, computes the high-level action the robot should perform at each step
  - But that action may have no feasible motion plan (recall: cluttered table)
- Motion planning – given a motion planning problem, computes the path that the robot should take
  - But which motion planning problem should it solve?

(Trajectory planning – selects the control inputs that should go to the robot's motors)



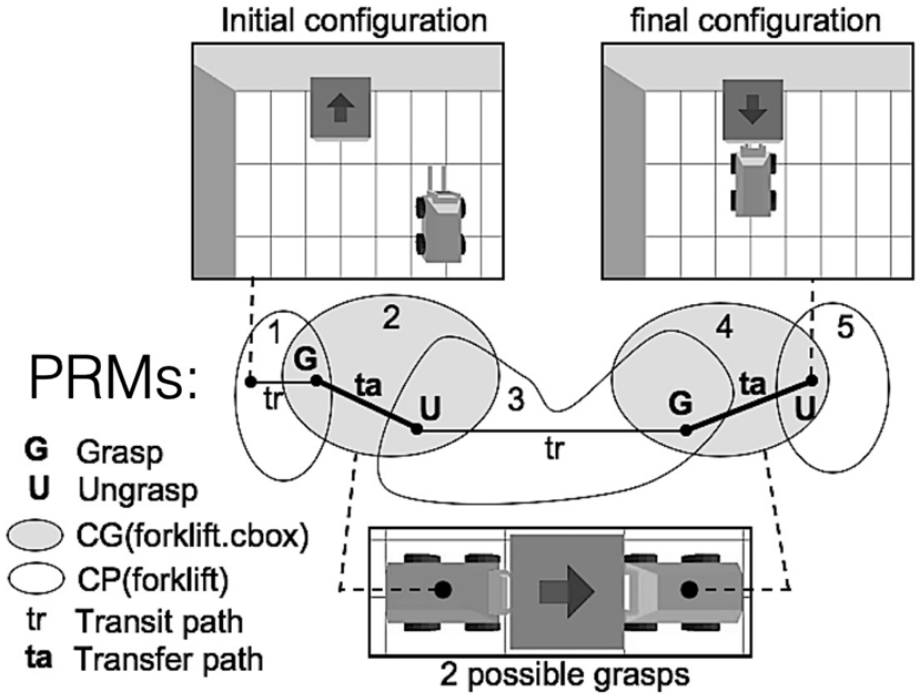
Early Characterization of Key Challenges: aSyMov

- Approach built upon PRMs
- *Articulated key technical problem:* picking and placing objects are discrete events, change the "robot", c-space
  - Picking an object leads to a new, composed robot
- Different composed versions of the robot have different C-spaces
- Solution Idea:
  - Maintain separate (projected) PRMs for different versions of the robot
  - Link them up based on actions such as pick and place
- Motion planning after a pick-up would use the PRM for the composed robot
- Where does task planning come in?
  - Goal specified in PDDL-like language
  - Task plan is used as a heuristic in PRM expansion

Cambon, S., Alami, R., & Gravot, F. (2009). A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*

aSyMov: Example Problem

- PRM for robot + box is disconnected
  - Components correspond to different grasps
- PRM for robot alone is also disconnected.
  - Components correspond to different box positions
- Need a path through linked points
- Grow a PRM per action; bias expansion by using high-level plan cost as a heuristic for c-states
- High-level model may be abstract (inaccurate) but used in a limited manner – not updated

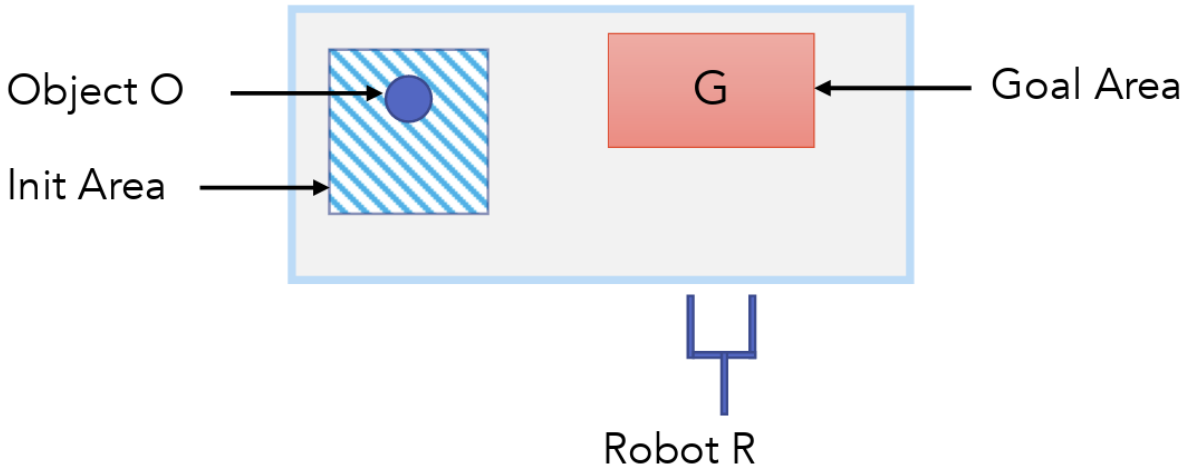


High-Level Summary

	Search Space	High Level Reasoning	Low Level Reasoning	High-level	High Level Language
aSyMov	Single	Any TP	PRM/RRT	Symbolic	PDDL

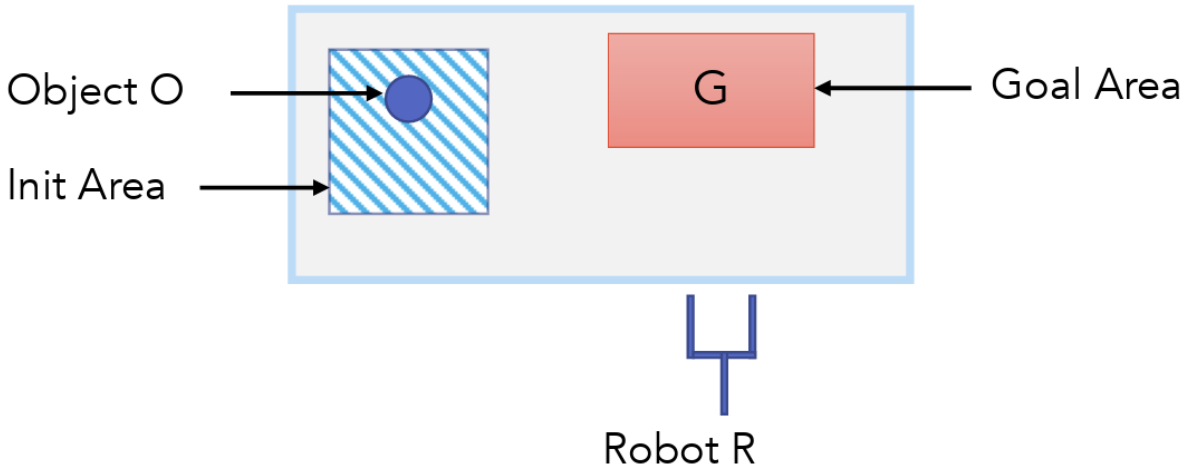
Domain for Robot Planning

```
(:action Move
 :parameter ?robot ?location ?trajectory
 :precondition
       not At(?robot ?location)
       Collision-free(?trajectory)
 :effect
       At(?robot ?location)
 )
```



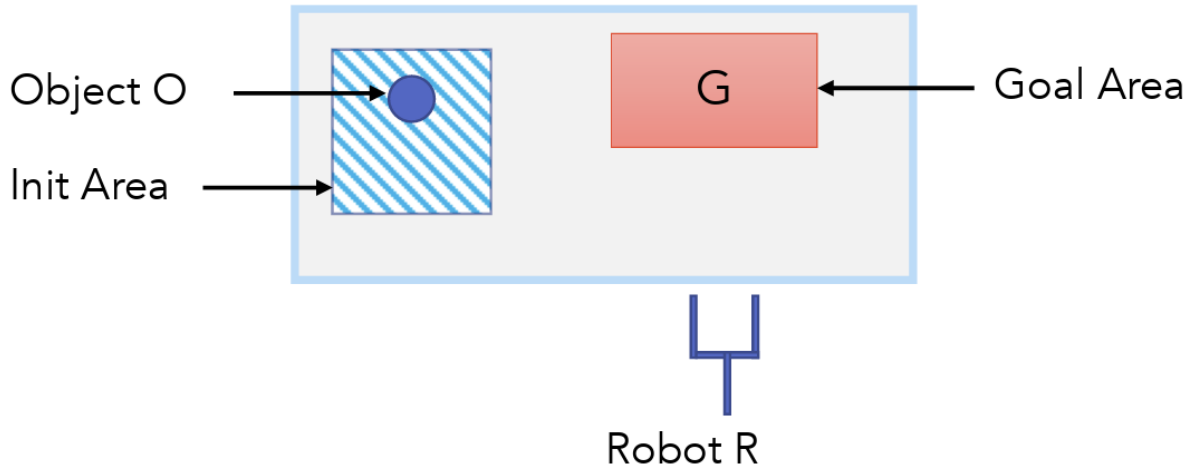
Domain for Robot Planning

```
(:action Move
 :parameter ?robot ?location ?trajectory
 :precondition
       not At(?robot ?location)
       Collision-free(?trajectory)
 :effect
       At(?robot ?location)
 )
```



Key challenges: infinitely many facts, infinite branching factor

Abstraction: State Abstraction



E.g., At(O,Init) is True iff $\exists l \in BlueArea$ s.t. pose(o,l) = True.

Abstraction: State Abstraction

First-order logic queries from concrete vocabulary V_l to abstract vocabulary V_h where $V_h \subset V_l$.

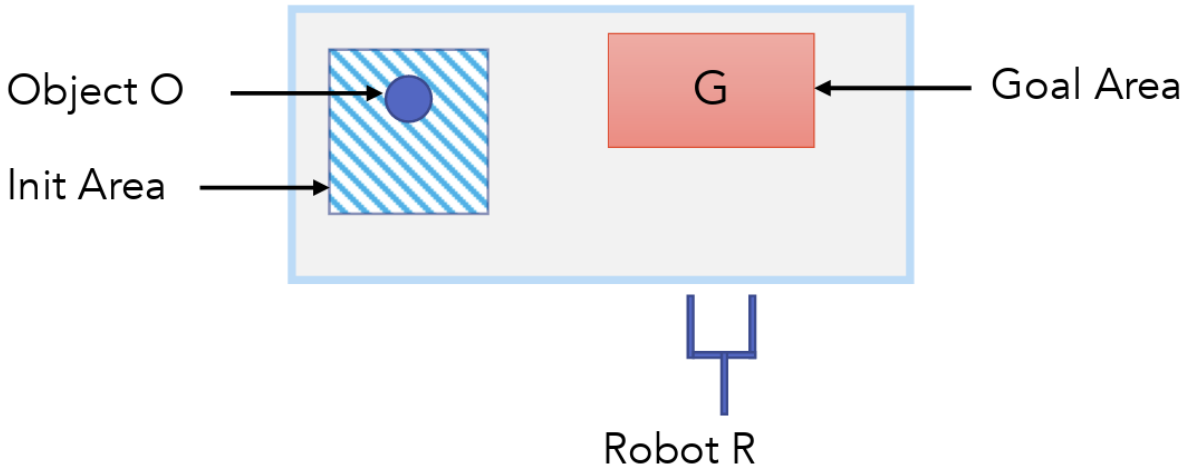
The query $[r]_{s_h}(\overline{o_1}, \ldots, \overline{o_n}) = True$ iff

$\exists o_1, \ldots, o_n$ such that $o_i \in \rho(\overline{o_i})$ and

$\left[\varphi_r^{\alpha_\rho}(o_1, \ldots, o_n)\right]_{S_l} = True$.

Collection function

Symbolic Reference



E.g., At(O,Init) is True iff $\exists l \in BlueArea$ s.t. pose(o,l) = True.

Here, $\rho(Init) = BlueArea = \{p_1, \ldots, p_n\}$

Abstraction: Symbolic Actions

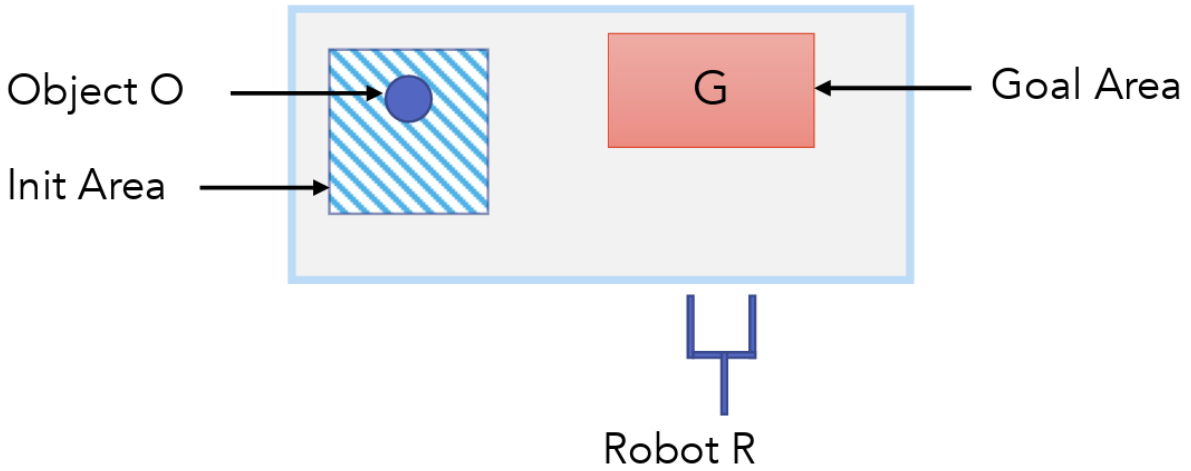
First-order logic queries from concrete vocabulary V_l to abstract vocabulary V_h where $V_h \subset V_l$.

The query $[r]_{s_h}(\overline{o_1}, \ldots, \overline{o_n}) = True$ iff

$\exists o_1, \ldots, o_n$ such that $o_i \in \rho(\overline{o_i})$ and

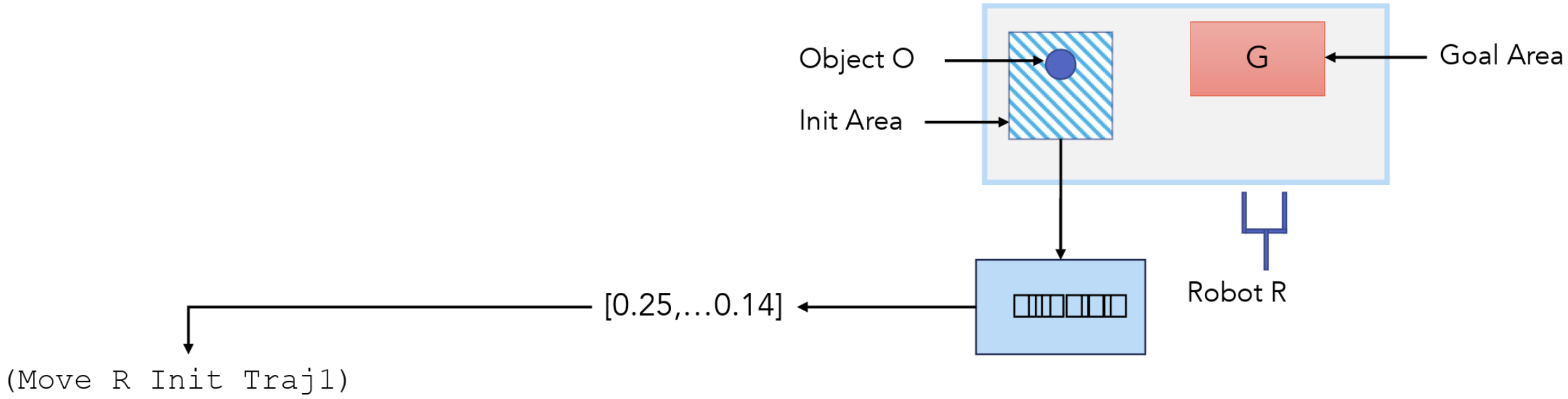
$\left[\varphi_r^{\alpha_\rho}(o_1, \ldots, o_n)\right]_{S_l} = True$.

```
(:action Move
 :parameter ?robot ?location ?trajectory
 :precondition
       not At(?robot ?location)
       Collision-free(?trajectory)
 :effect
       At(?robot ?location)
 )
```

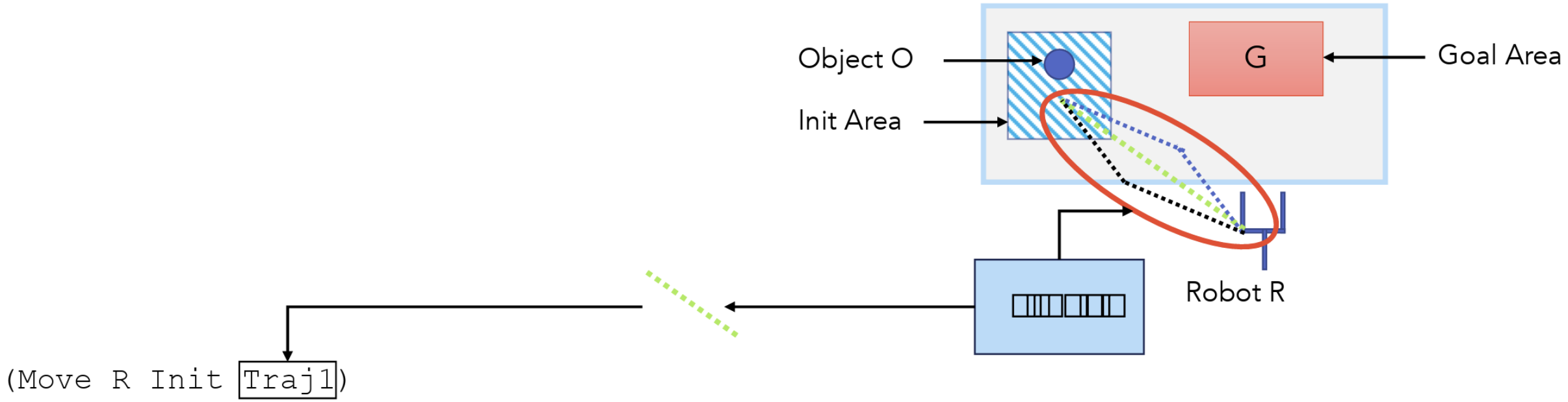


Symbolic arguments that can be instantiated with symbolic references for high-level planning

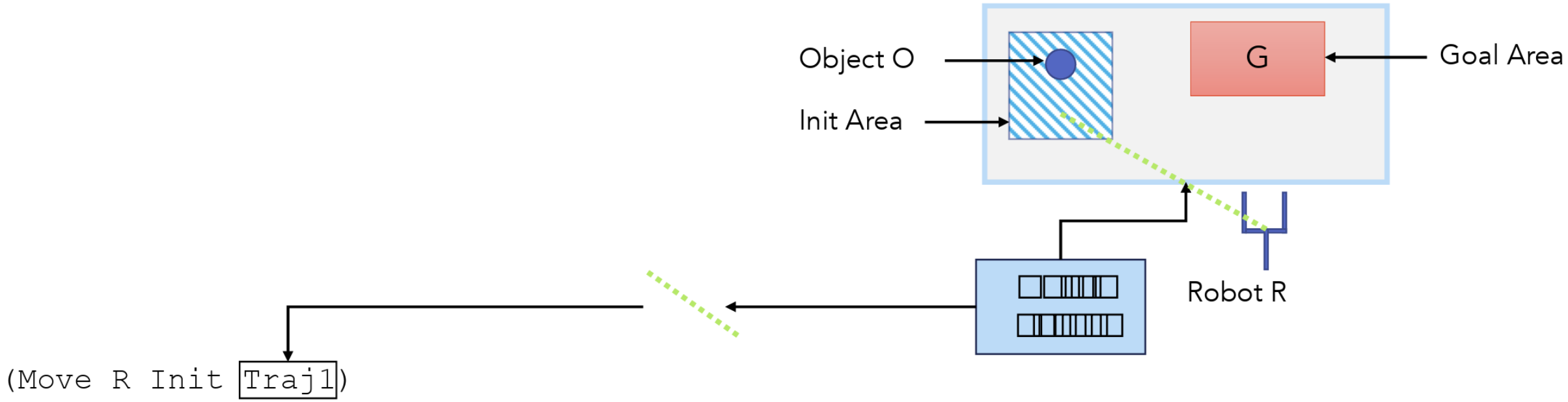
Abstraction: State Abstraction



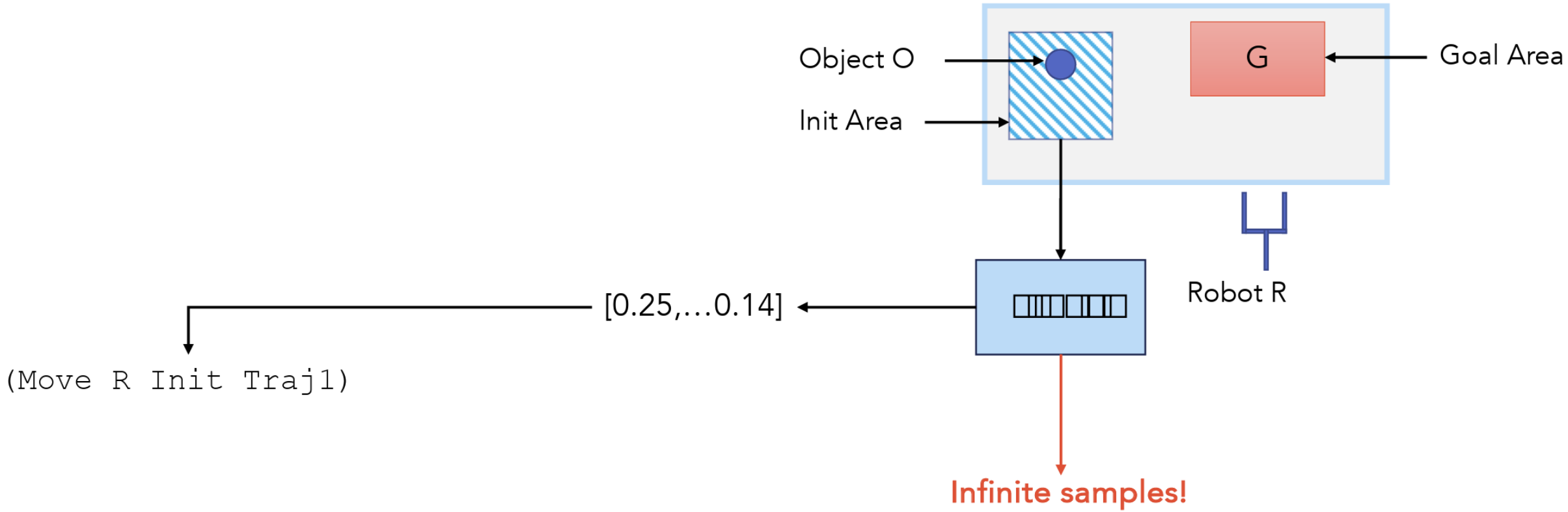
Abstraction: Refining Symbolic Action



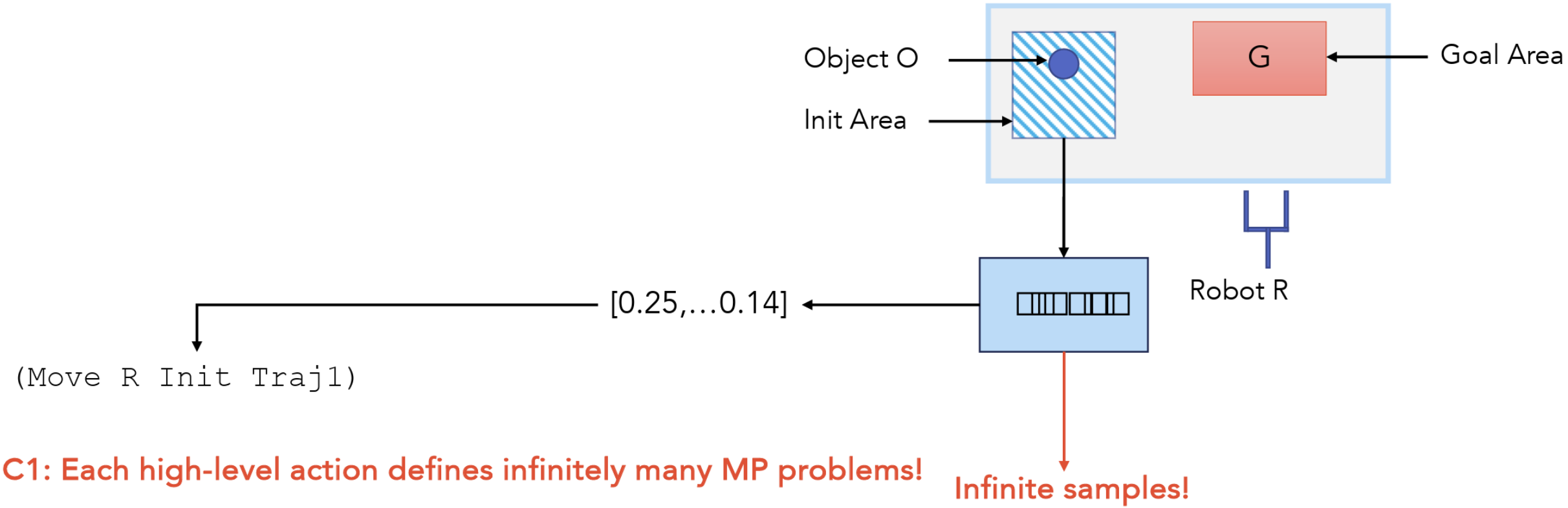
Abstraction: State Abstraction



Challenge 1 – Which MP to Solve?



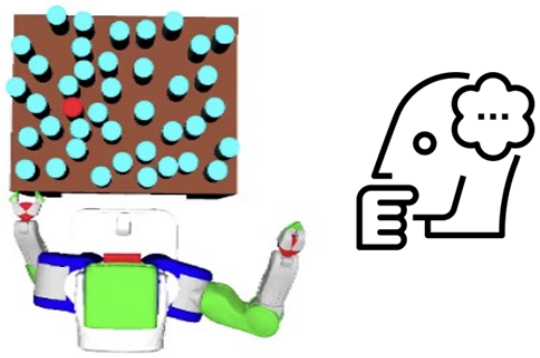
Challenge 1 – Which MP to Solve?



Challenge 2 – Which MP Will be Solvable?

Can pickup only from the side

Obstructions depend on choice of movement trajectory



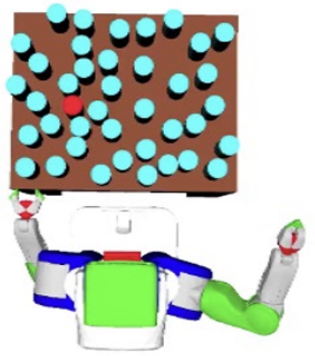
No way to verify which trajectories are collision-free

```
(:action Move
 :parameter ?robot ?location ?trajectory
 :precondition
       not At(?robot ?location)
       Collision-free(?trajectory)
 :effect
       At(?robot ?location)
 )
```

Challenge 2 – Which MP Will be Solvable?

Can pickup only from the side

Obstructions depend on choice of movement trajectory



No way to verify which trajectories are collision-free

```
(:action Move
 :parameter ?robot ?location ?trajectory
 :precondition
       not At(?robot ?location)
       Collision-free(?trajectory)
 :effect
       At(?robot ?location)
 )
```

C2: Loss of information in abstraction → Actions in plan that can not refinable

Task and Motion Planning Problem

O, universe of objects and object poses (implicitly defined)

$P = P_{sym} \cup \bar{P}_h$, set of symbolic predicates and interpretations (definitions in geometric constraints)

Generates S, set of abstract states

$A = A_{sym} \cup \bar{A}_h$, set of abstract actions

$T: S \times A \rightarrow \mu S$, action transition function

$R: S \times A \rightarrow \mathbb{R}$, costs and utility of states, actions (can express goals and some forms of preferences*)

γ, a concretization function (often in the form of samplers or generators)

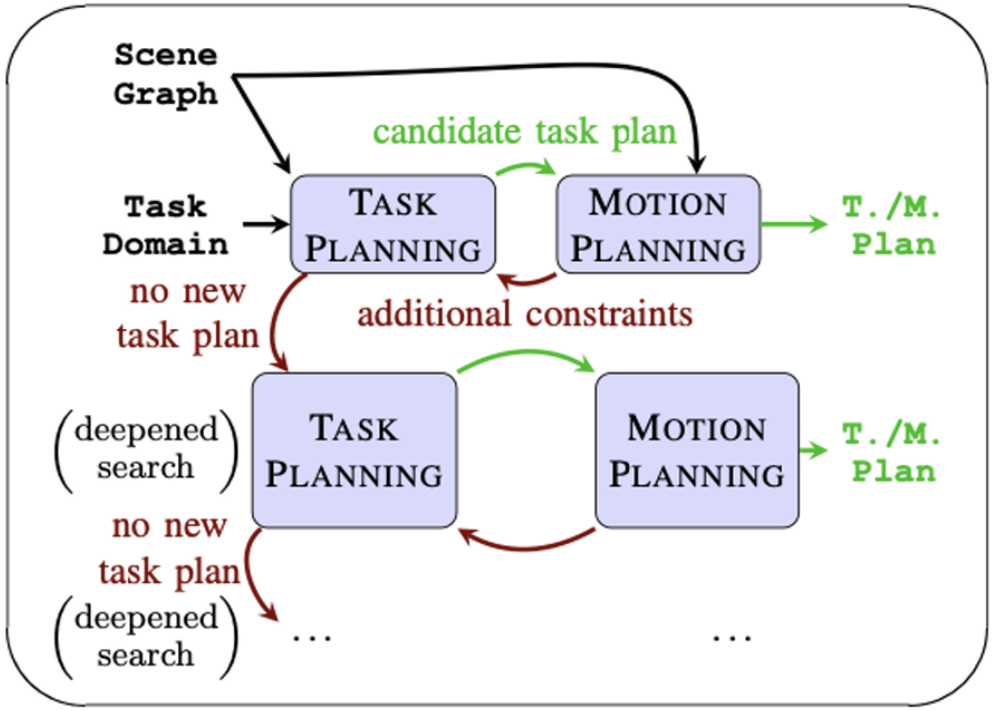
Task and motion planning:

Compute a sequence of actions from A that maximizes the utility R and that can be executed in the given model.

Optimization and SMT Based Approaches

IDTMP

- Core idea: Use advances in SAT/SMT solvers to perform hybrid search for TMP
- State variables for different objects and use poses as values
- Convert each high-level action to low-level motion planning problem.



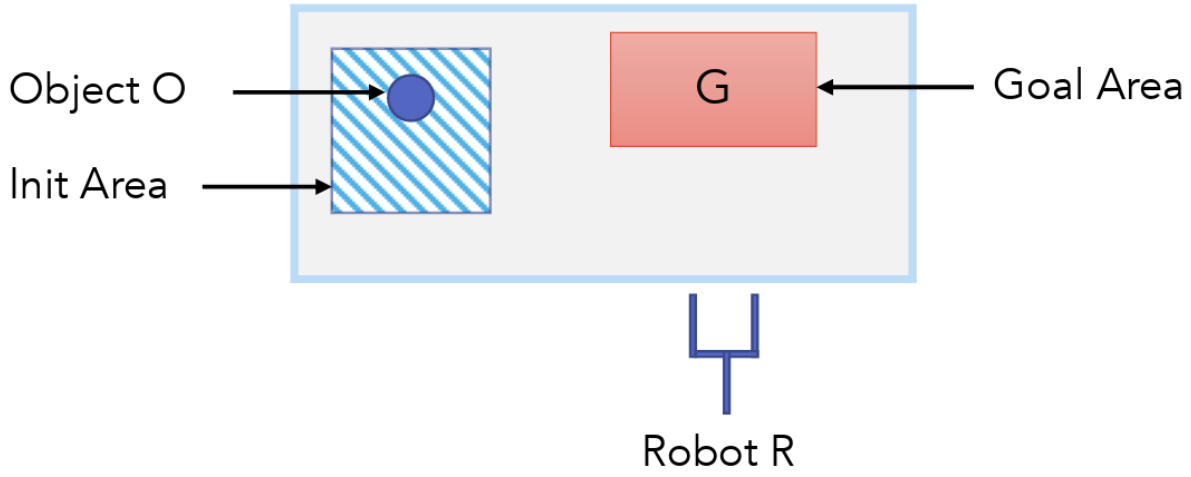
Nedunuri, Srinivas, et al. "SMT-based synthesis of integrated task and motion plans from plan outlines." *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

IDTMP: SMT Planning

- SMT = satisfiability module theories
- Model the problem as Boolean satisfiability problem

State variables:
Pose of the object $P_0 \in R^2$
Configuration of the robot $C_R \in \mathcal{X}$

Actions:
Pick
Place
Move

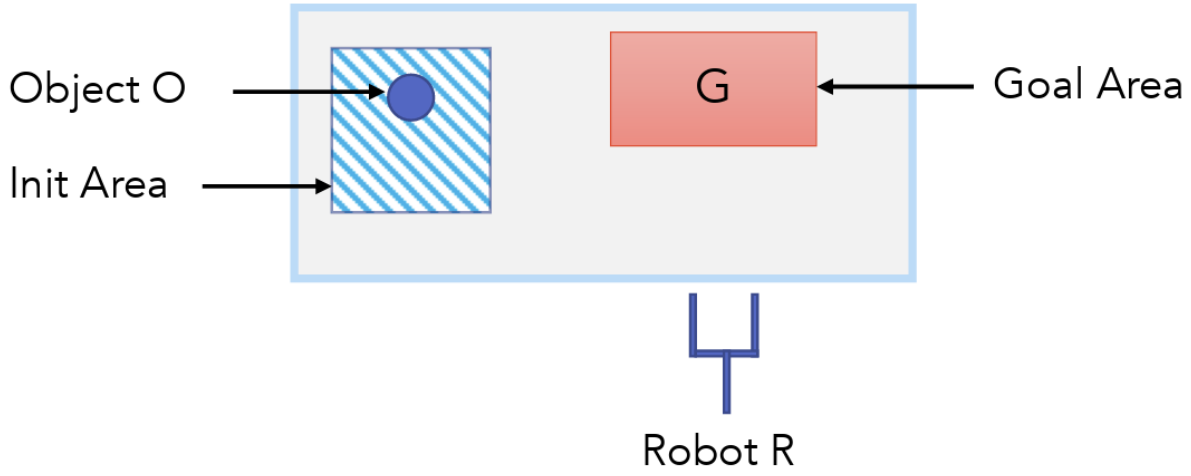


IDTMP: SMT Planning

State variables:
Pose of the object $P_0 \in R^2$
Configuration of the robot $C_R \in \mathcal{X}$

Actions:
Pick
Place
Move

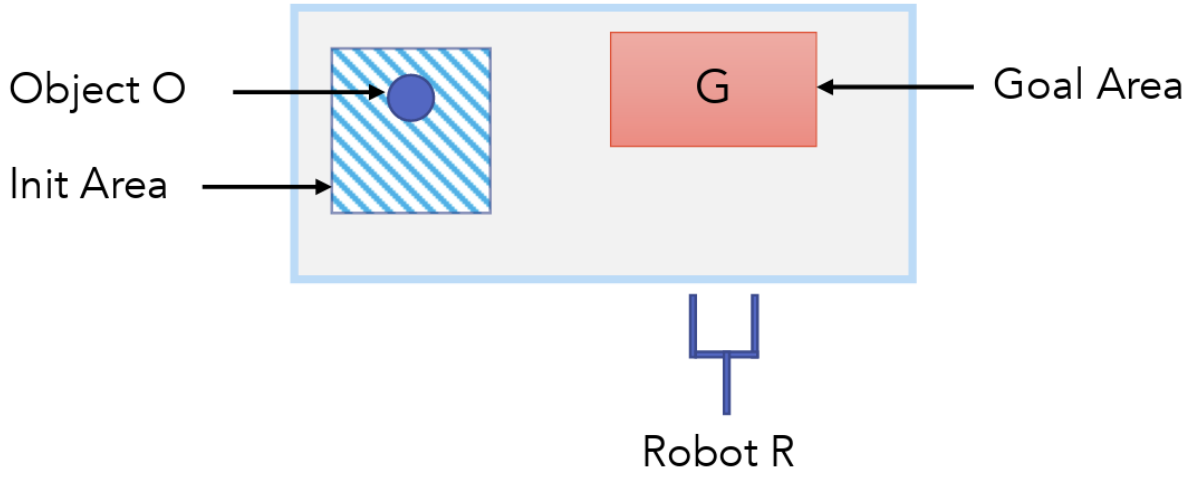
$P = \{p_0, \ldots, p_n\}$ (a set of state variables)
$A = \{a_0, \ldots, a_k\}$ (a set of actions)



IDTMP: SMT Planning

State variables:
Pose of the object $P_0 \in R^2$
Configuration of the robot $C_R \in \mathcal{X}$

Actions:
Pick
Place
Move



$P = \{p_0, \dots, p_n\}$ (a set of state variables)

$A = \{a_0, \dots, a_k\}$ (a set of actions)

Three constraints:

1. $a_i^k \Rightarrow Pre(a_i)^k \wedge Eff(a_i)^{k+1}$ // If an action is taken at the step k, the its precondition and effect must hold
2. $\left(p_i^k = p_i^{k+1}\right) \vee (a_j^k \vee \dots \vee a_l^k)$ // Variables that are not changed by the actions remains unchanged
3. $a_i^k \Rightarrow \neg\,(\,a_0^k \vee \dots a_{i-1}^k \vee \dots a_{i+1}^k \vee \dots a_l^k)$ // Only one action can be taken at the given step

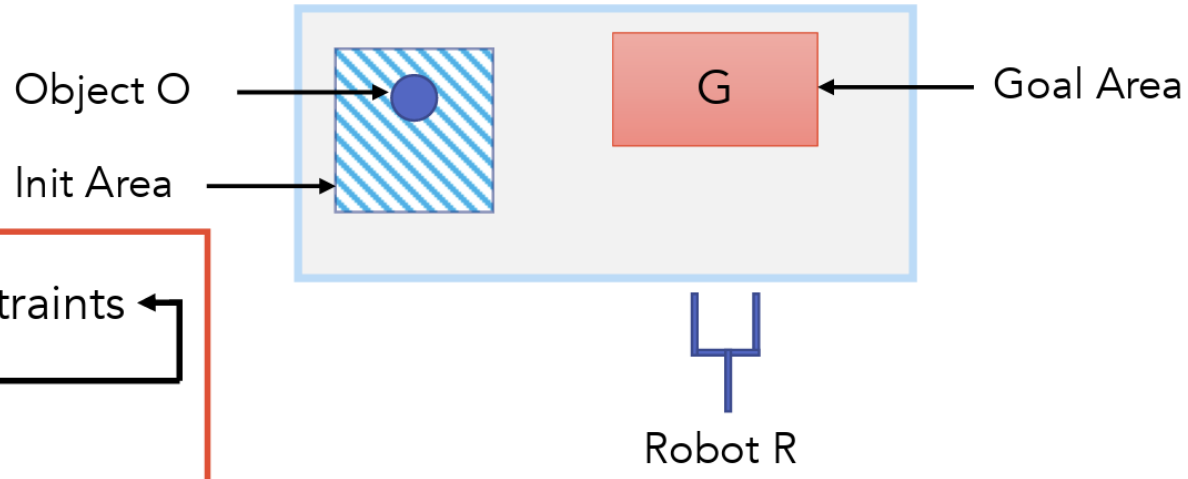
SMT formula using p_i and a_i // $p_i \in P$ (a set of state variables) and $a_i \in A$ (a set of actions)

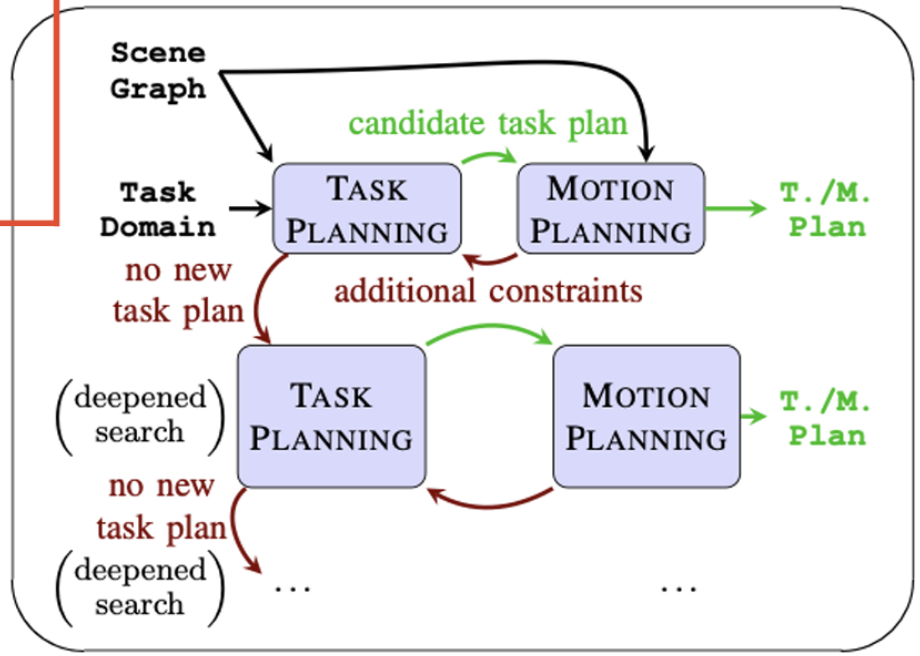
SMT used
1) solve SMT formula consisting of discrete action and Boolean variables
2) maintain dynamic constraints such as action a_i is not applicable at step t

IDTMP: SMT Planning

- Plan_length = 1
- Constraints = initial constraints
- Compute task plan for the current plan length and constraints
- If no task plan found: plan_length + 1 and retry
- If a task plan is found:
  - For every action in task plan:
    - Compute a motion plan
      - If no motion plan:
        - Add new constraints

If a_i^k does not have a motion plan → disallow a_i at step k.





High-Level Summary

	Search Space	High Level Planner	Low Level Reasoning	High-level	High Level Language	P1: Infinite Motion Plans	P2: Downward Refinability
aSyMov	Single	Any TP	PRM/RRT	Symbolic	PDDL	N/A	N/A
IDTMP	Dual	SMT	Any MP	Variables with continuous domains	SAS	SMT	SMT

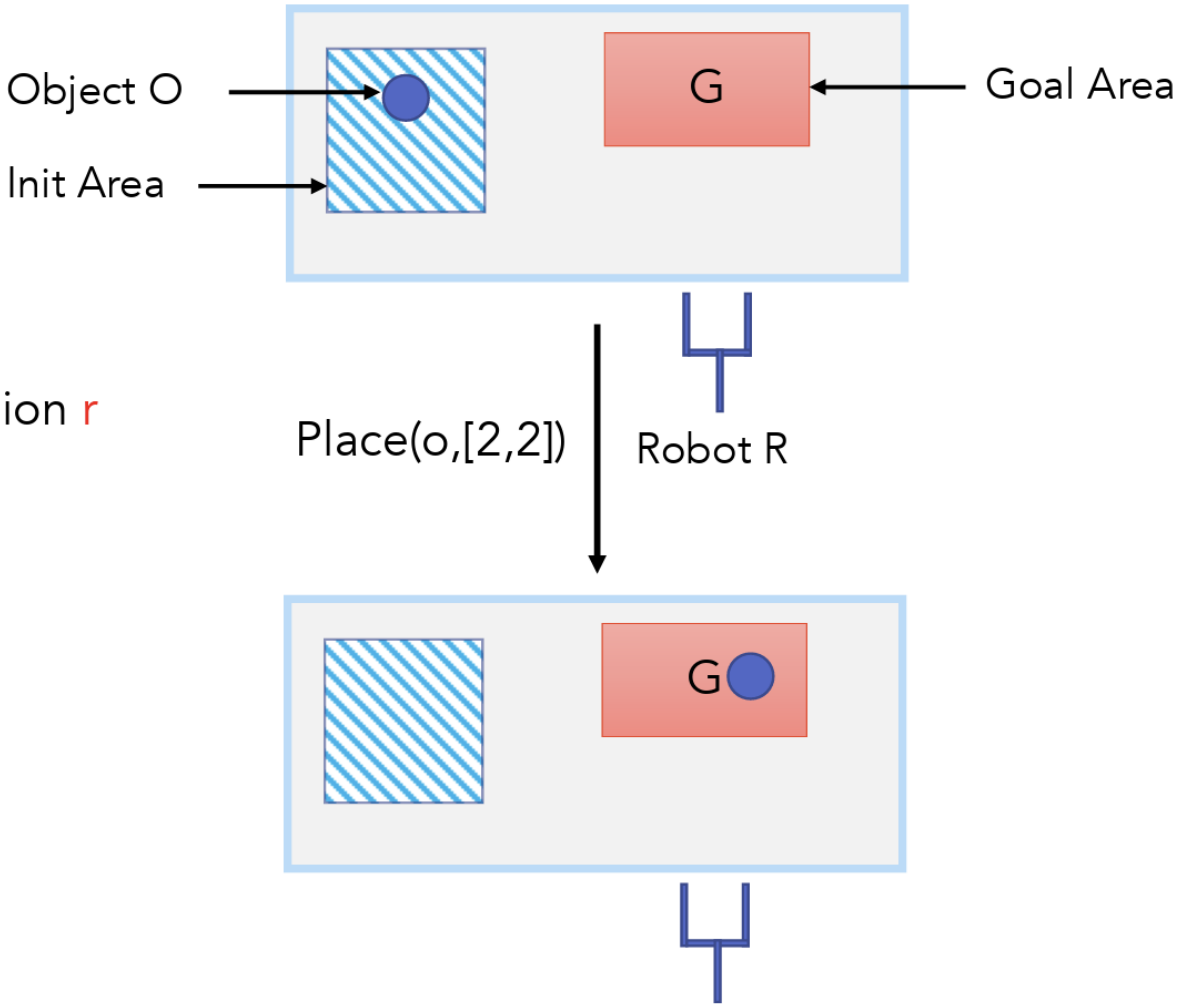
Hierarchical Planning in the Now: HPN

Central Idea:

- Don't abstract the state; plan over fluents & actions with continuous arguments
- For high-level reasoning: Use regression of geometric fluents
- For efficiency: Use an operator-abstraction hierarchy
- For refinement: Interleave refinement with execution

HPN: States and Actions

- State represented by a set of fluents with possibly continuous arguments
  - Subroutines used to dynamically evaluate fluents
  - No need to represent complete state descriptions
- Examples of fluents with continuous arguments (1-d):
  - In(o, r): object o is completely inside region r
  - ObjLoc(o, l): left edge of object o is at location l
  - ClearX(r, x): only objects ∈ x possibly overlap with region r
- Actions:
  - **Place**(o, l_{target}) causes ObjLoc(o, l_{target});
    - requires ClearX(sweptVol(o, l_{init}, l_{target}))
  - **In**(o, r): ramification action for the fluent In(o, r)
  - **Clear**(r, x): ramification action for the fluent Clear(r, x)



HPN: Action Specifications

- Action specification without hierarchy:
  - Arguments include current subgoal (maintained during regression)
  - Effects, preconditions including argument-choice constraints

```
PickPlace((o, l_target ), s_now , γ):
        effect: ObjLoc(o, l_target)
        choose: l_start ∈ {s_now[o].loc} ∪
                generateLocsInRegions((o,{warehouse,stove,sink}),s_now,γ)
                //operator instantiations have to be considered for each generated value of l_start
                //Allows l_start to be generated using γ for efficient regression
        pre: ObjLoc(o, l_start), ClearX(sweptVol(o, l_start,l_target),{o})
```

- Precondition + choose essentially generates the next subgoal during regression
- Use operator-specific regression subroutines for geometric fluents (provided as input)

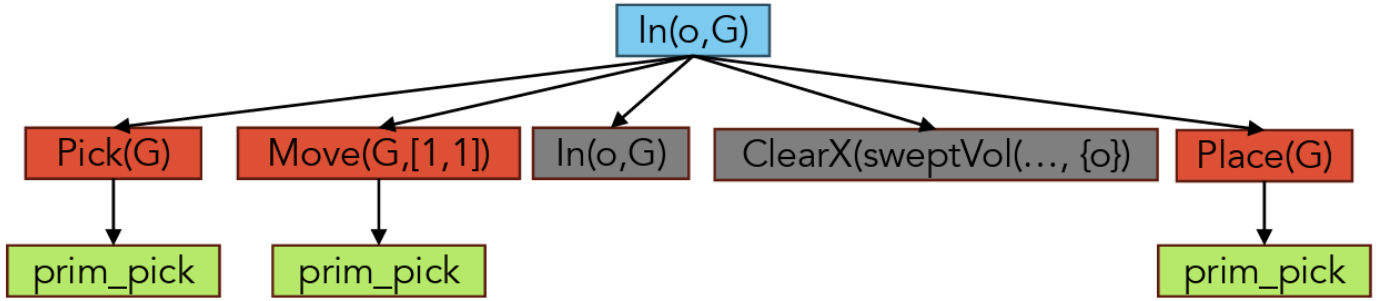
HPN: Regression Algorithm

- Carry out goal regression using goal state, preimage computation methods for each operator, geometric fluent
- Search using A*; heuristic = number of goal fluents that are not true in the preimage

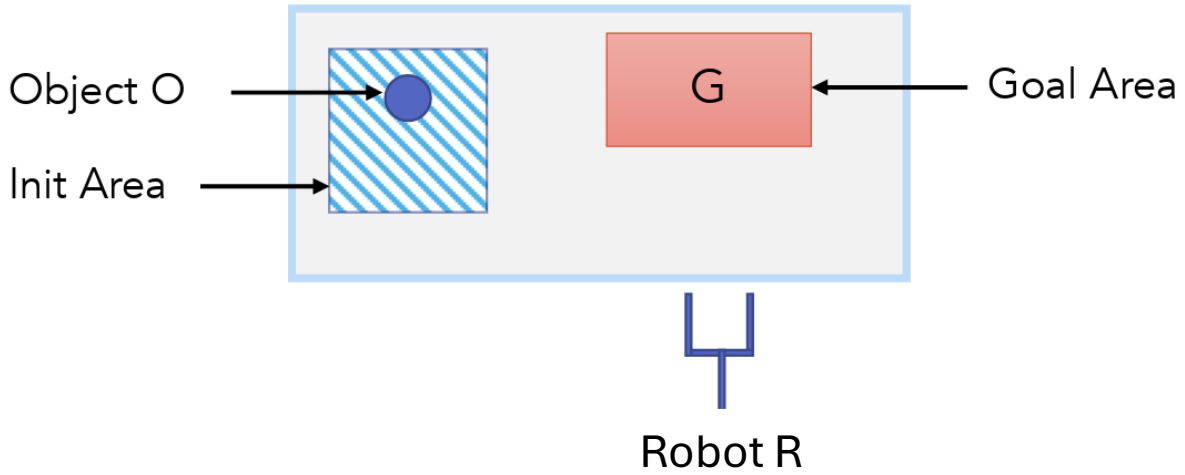
Pick((o),s,γ):
  effect: Holding(o)
  pre: AtGraspPose(o) [1]
  prim: prim_pick

Place((o),s,γ):
  effect: Holding(o) [1]
  pre: ¬Holding(o)
  prim: prim_place

Move((o,l_{end}),s,γ):
  choose: l_{start}
  effect: ObjectLoc(l_{end})
  pre: ObjectLoc(l_{start}) [1]
    Holding(o) [2]
    ClearX(sweptVol(o, l_{start}, l_{end}), {o}) [3]
  prim: prim_move



Goals
Operations
Ramification actions
Primitive actions



HPN: Regression Algorithm

- Use a hierarchy defined using precondition postponement to reduce the horizon during regression
- Suppose operator o has preconditions p_1, …, p_n, effect r
- Operator with p_n postponed
  - o_postponed:
    - precon = p_1, …, p_{n-1}
    - expansion:
      - Achieve p_n while maintaining p_1, …, p_{n-1};
      - Then execute o
- Additional side-effects of achieving p_n may have to be declared
- Define hierarchy by associating abstraction-level with each precondition

Pick((o),s,γ):
effect: Holding(o)
pre: AtGraspPose(o) [1]
prim: prim_pick

Place((o),s,γ):
effect: Holding(o) [1]
pre: ¬Holding(o)
prim: prim_place

Move((o,l_{end}),s,γ):
choose: l_{start}
effect: ObjectLoc(l_{end})
pre: ObjectLoc(l_{start}) [1]
Holding(o) [2]
ClearX(sweptVol(o, l_{start}, l_{end}), {o}) [3]
prim: prim_move

HPN: Overall Approach

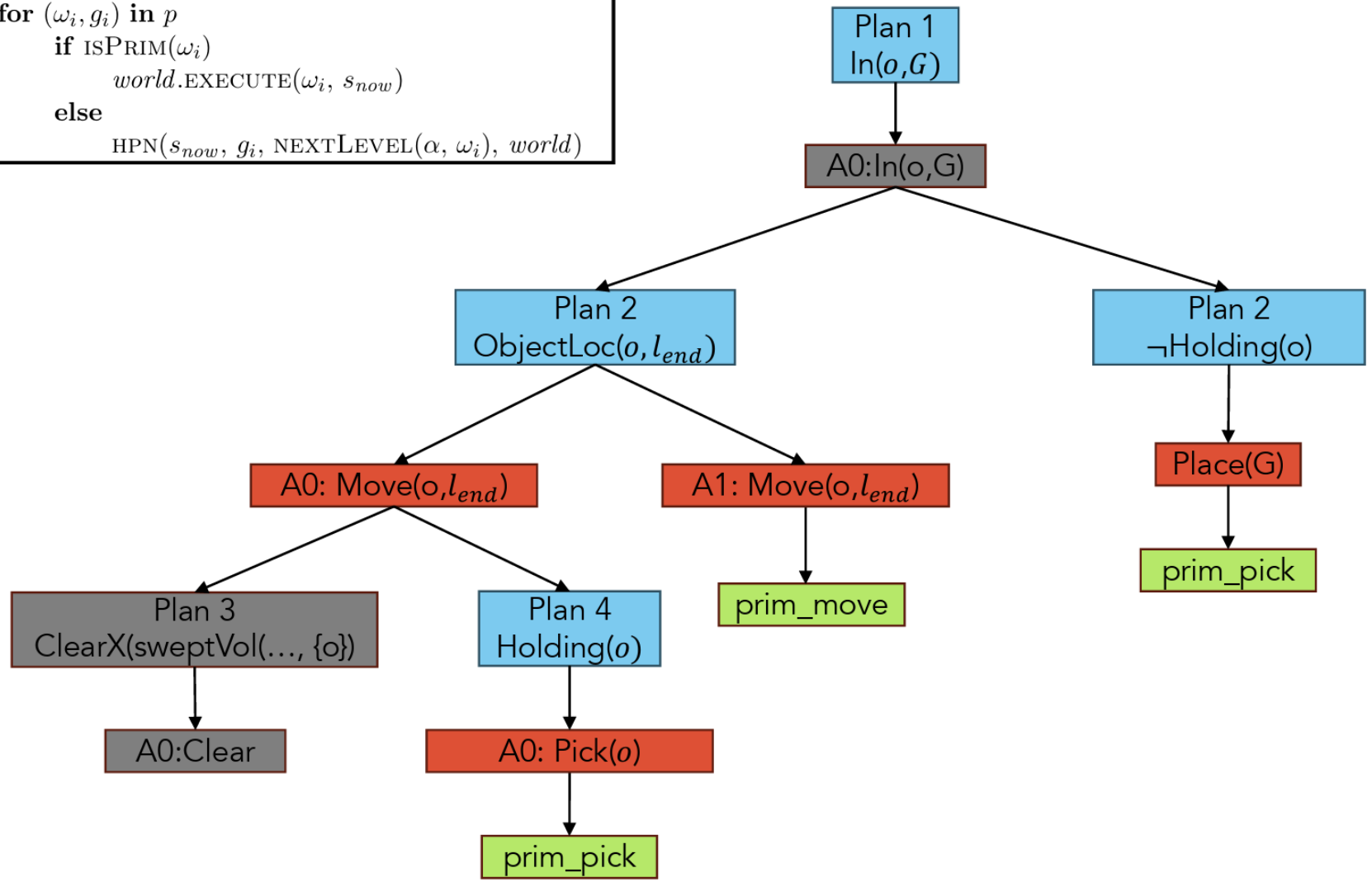
Pick((o),s,γ):
- effect: Holding(o)
- pre: AtGraspPose(o) [1]
- prim: prim_pick

Place((o),s,γ):
- effect: ¬ Holding(o) [1]
- pre: Holding(o)
- prim: prim_place

Move((o,l_{end}),s,γ):
- choose: l_{start}
- effect: ObjectLoc(l_{end})
- pre: ObjectLoc(l_{start}) [1]
  - Holding(o) [2]
  - ClearX(sweptVol(o, l_{start}, l_{end}), {o}) [3]
- prim: prim_move

In((o,R),s,γ):
- choose: l
- effect: In(o, R)
- pre: ObjectLoc(l) [1]
  - ¬Holding(o) [2]

```
HPN(s_now, γ, α, world):
    p = PLAN(s_now, γ, α)
    for (ω_i, g_i) in p
        if ISPRIM(ω_i)
            world.EXECUTE(ω_i, s_now)
        else
            HPN(s_now, g_i, NEXTLEVEL(α, ω_i), world)
```



HPN: Summary

- Input:
  - Correct and complete primitive action definitions using geometric predicates
  - Operator-specific regression functions for geometric properties
  - Pose generators
  - For efficiency, can use additional input:
    - Pose generators that make use of current subgoal
    - Precondition levels to obtain a hierarchy, declaration of operator side-effects
    - Limited logical reasoning in pose generators
- Properties:
  - Complete if the problem has no dead-ends, action preconditions are accurate
    - Motion planners terminate and return solutions when preconditions hold
- Approach for C1: through generators and regression (backward-search)
- Approach for C2: Regression over geometric fluents

Kaelbling, Leslie Pack, and Tomás Lozano-Pérez. "Hierarchical task and motion planning in the now." *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.

High-Level Summary

	Search Space	High Level Planner	Low Level Reasoning	High-level	High Level Language	P1: Infinite Motion Plans	P2: Downward Refinability
aSyMov	Single	Any TP	PRM/RRT	Symbolic	PDDL	N/A	N/A
IDTMP	Dual	SMT	Any MP	Variables with continuous domains	SAS	SMT	SMT
HPN	Dual	HPN-specific regression planner	Any MP	Hybrid	HPN-specific	generators	Regression over geometric fluents

TMP Through an Interface Layer

- Geometric planning is hard → Symbolic high-level
- Off-the-shelf task planner
- Off-the-shelf motion planner
- Interface layer that communicates between task planner and motion planner
  - Converts each task level action to a motion planning problem
  - Converts motion planning failures as facts over symbols & refines abstract information

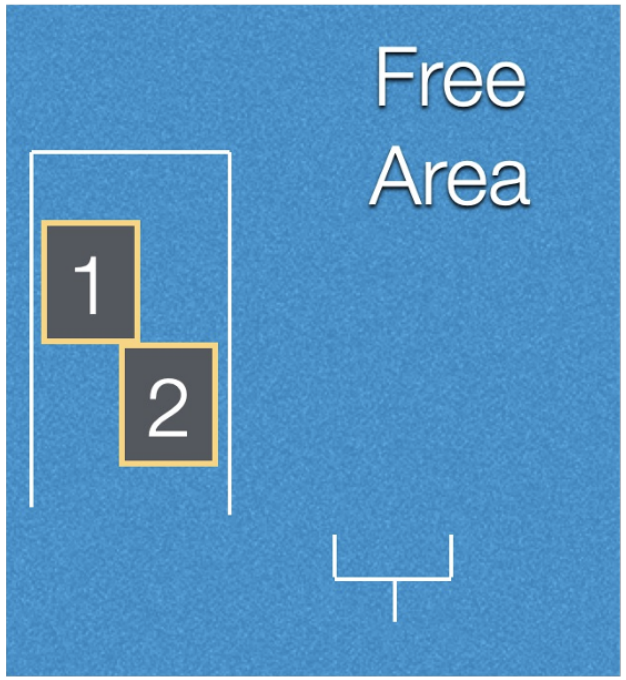
Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., & Abbeel, P. (2014, May). Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*

TMP Through an Interface Layer: Example

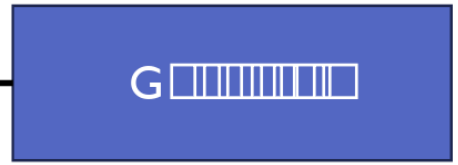
PickUp(obj o1, pose p1, pose p2, pose p3, path p):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)



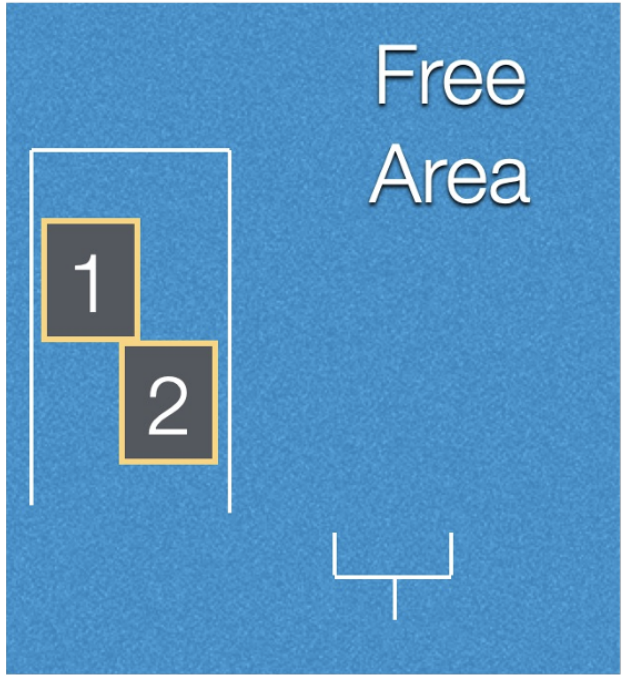
TMP Through an Interface Layer: Example



PickUp(obj o1, pose p1, pose p2, pose p3, path p):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)



TMP Through an Interface Layer: Example

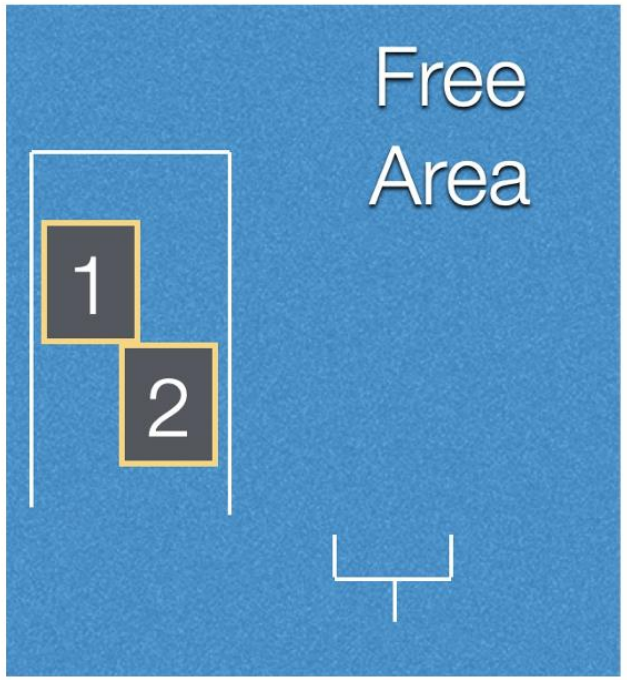
PickUp(obj o1, **pose p1, pose p2, pose p3, path p**):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)

Symbolic references

- High level intuitive plan:
  - pick block1 after going to *block1's grasping pose* along *a trajectory*



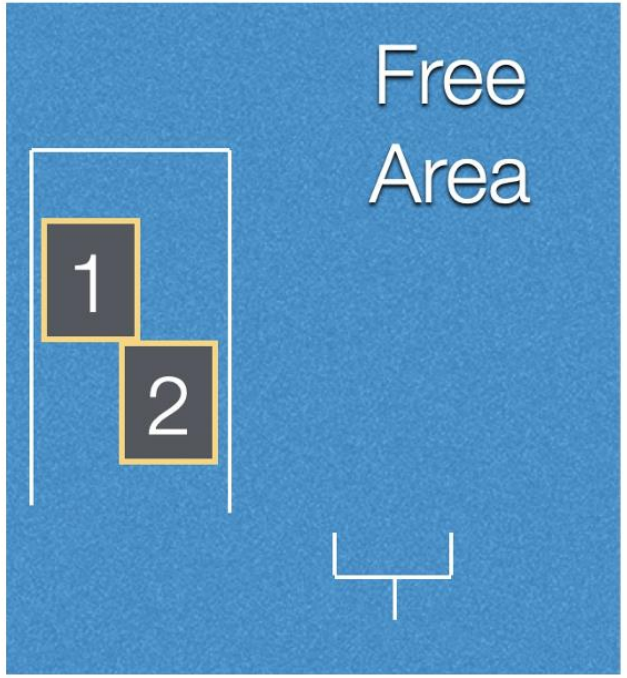
TMP Through an Interface Layer: Example

Interface level:
Searches for an instantiation of block1's grasping pose that is reachable via a feasible (collision-free) trajectory

…finds no feasible trajectory

Symbolic references

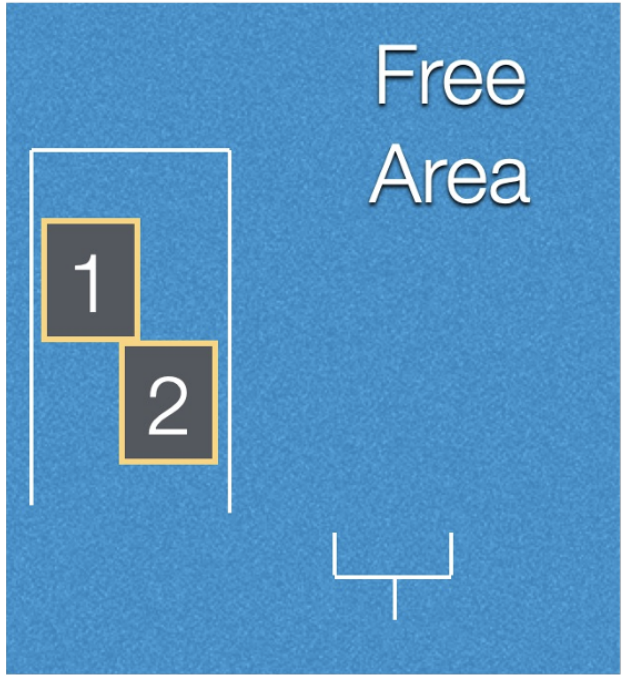
- High level intuitive plan:
  - pick block1 after going to *block1's grasping pose* along *a trajectory*



TMP Through an Interface Layer: Example

Interface level:
Searches for an instantiation of block1's grasping pose that is reachable via a feasible (collision-free) trajectory

…finds no feasible trajectory



Symbolic references

- High level intuitive plan:
  - ~~pick block1 after going to *block1's grasping pose* along *a trajectory*~~

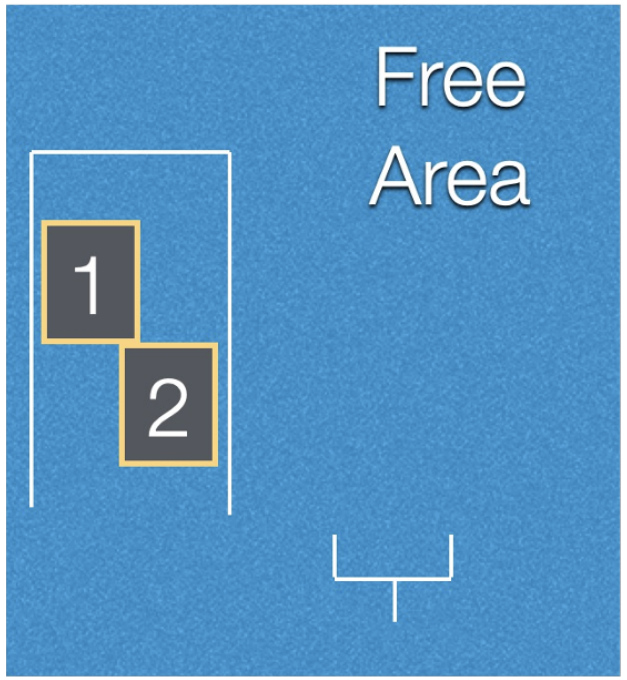
Fix values for references, report reason for failure:
"block2 obstructs *block1's grasping pose* along *a trajectory*"

TMP Through an Interface Layer: Example

PickUp(obj o1, pose p1, pose p2, pose p3, path p):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)



Symbolic references

- High level intuitive plan:
  - ~~pick block1 after going to *block1's grasping pose* along *a trajectory*~~

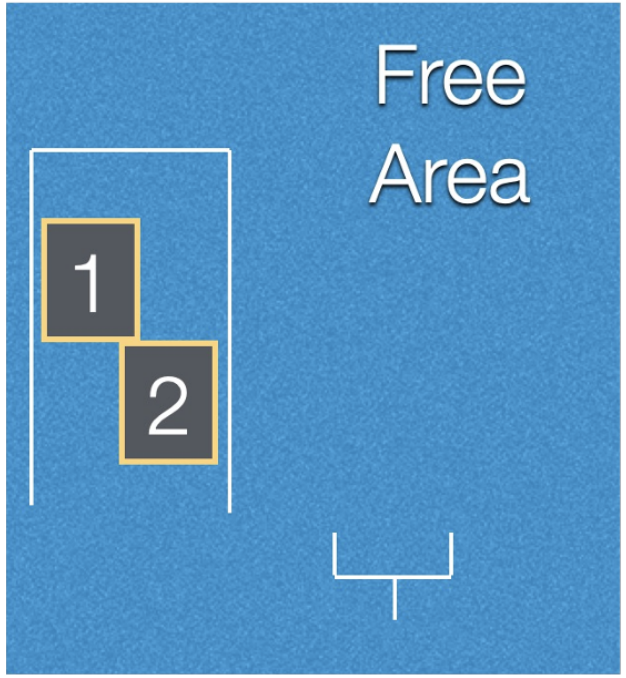
Fix values for references, report reason for failure:
"block2 obstructs *block1's grasping pose* along *a trajectory*"

TMP Through an Interface Layer: Example

PickUp(obj o1, pose p1, pose p2, pose p3, path p):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)



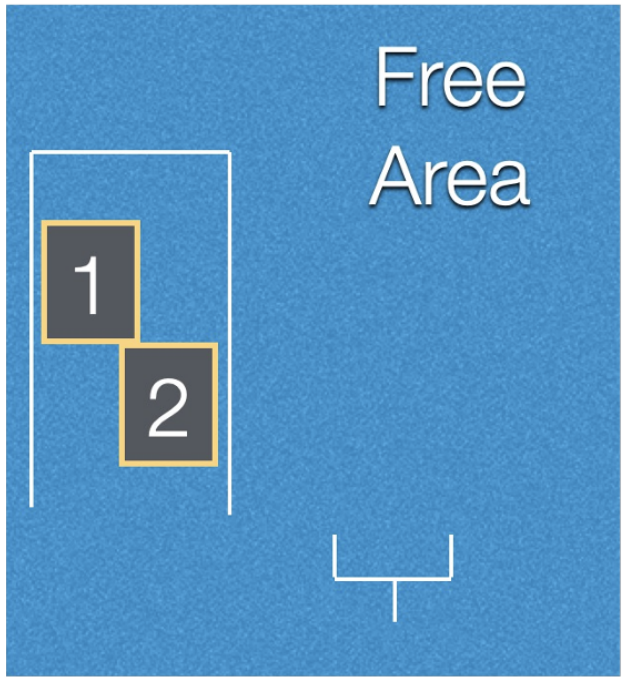
Discrete state += Obstructs(block2, path(initLoc, gp(block1))

TMP Through an Interface Layer: Example

PickUp(obj o1, **pose p1, pose p2, pose p3, path p**):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)



- High level intuitive plan:
  - ~~pick block1 after going to *block1's grasping pose…*~~
- REPLAN
  - pick block2 after going to *block2's grasping pose…*
  - release block2 after going to *release pose for free area…*
  - pick block1 after going to *block1's grasping pose…*

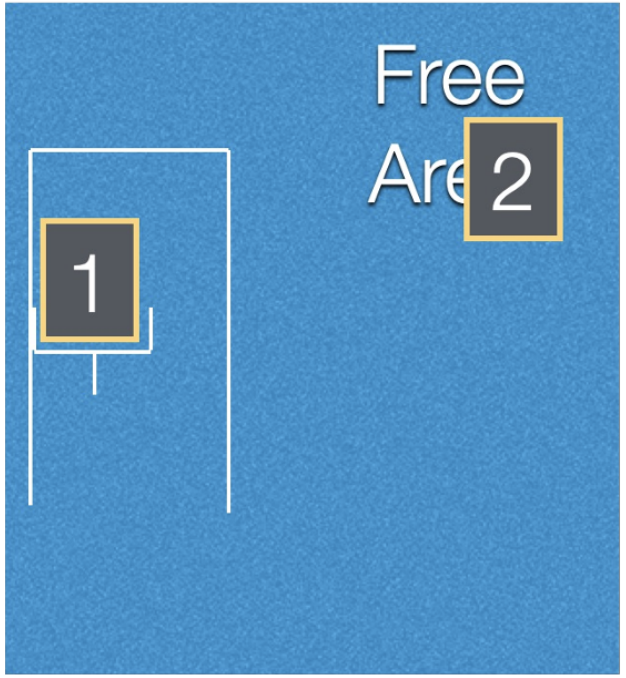
TMP Through an Interface Layer: Example

PickUp(obj o1, **pose p1, pose p2, pose p3, path p**):

precondition: Empty(gripper) ∧ GripperAt(p1) ∧
At(o1, p3) ∧ IsGraspingPose(p2, o1, p3)
∧ path(p, p1, p2) ∧ ∀o2 ¬ *Obstructs*(o2, p)

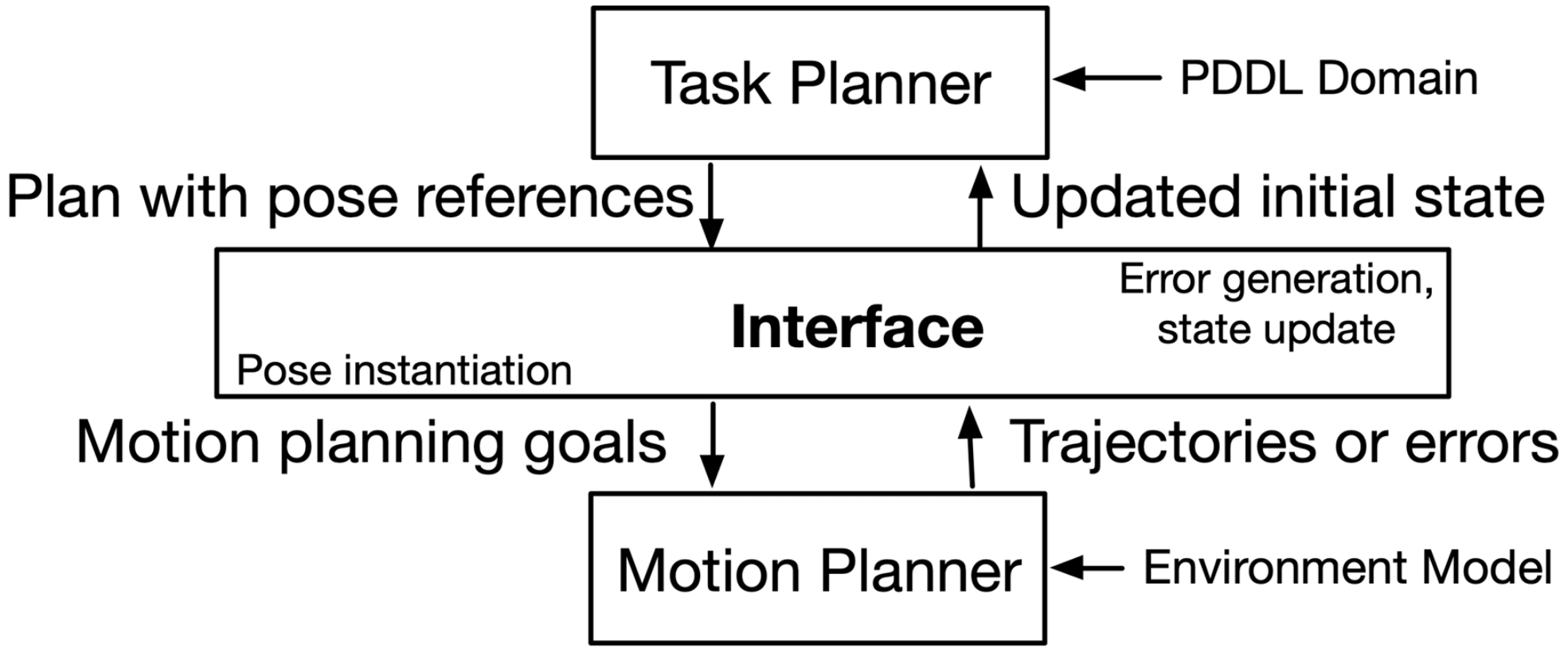
effect: Holding(o1) ∧ ¬ At(o1, p3) ∧
¬ Empty(gripper) ∧ GripperAt(p2)

- High level intuitive plan:
  - ~~pick block1 after going to *block1's grasping pose…*~~
- REPLAN
  - pick block2 after going to *block2's grasping pose…*
  - release block2 after going to *release pose for free area…*
  - pick block1 after going to *block1's grasping pose…*

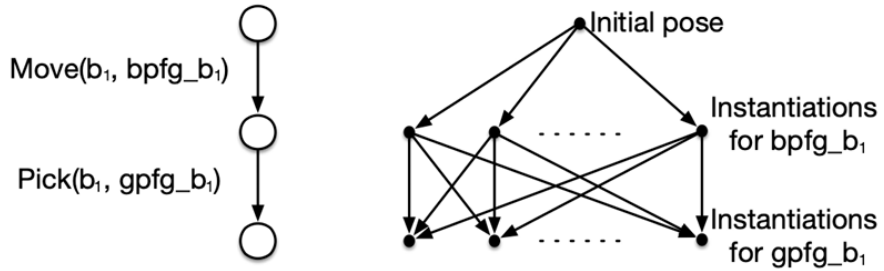


Goal Reached!!

TMP Through an Interface Layer: Complete Algorithm

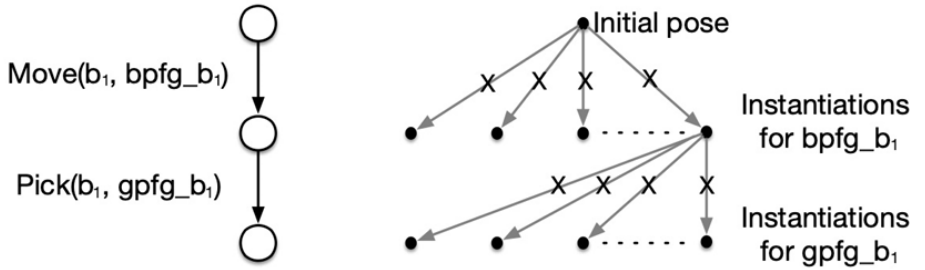


TMP Through an Interface Layer: Complete Algorithm

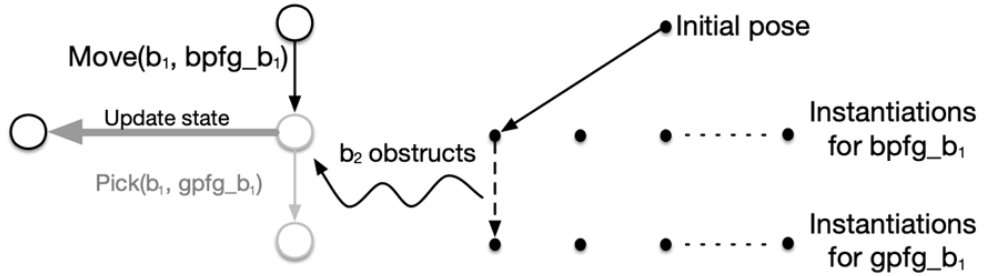


Iteratively try all instantiations

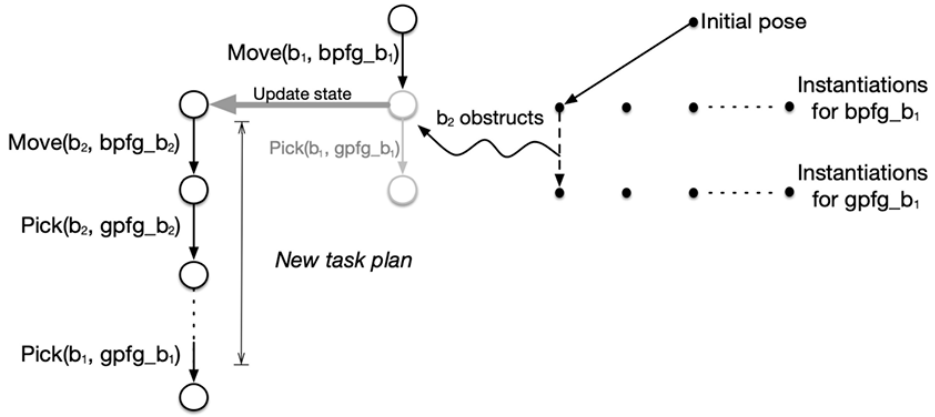
What if that fails?



Fix the symbolic state using failure reason



Compute a new plan



High-Level Approach

	Search Space	High Level Planner	Low Level Reasoning	High-level	High Level Language	P1: Infinite Motion Plans	P2: Downward Refinability
aSyMov	Single	Any TP	PRM/RRT	Symbolic	PDDL	N/A	N/A
IDTMP	IDTMP	Dual	SMT	Any MP	Variables with continuous domains	SAS	SMT
HPN	Dual	HPN-specific regression planner	Any MP	Hybrid	HPN-specific	generators	Regression over geometric fluents
Symbolic Interface	Dual	Any TP	Any MP	Symbolic	PDDL	Pose generators	Identifying errors