Reasoning for Robot Manipulation

Jan Rosell and Mohammed Diab

Institute of Industrial and Control Engineering (IOC)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)

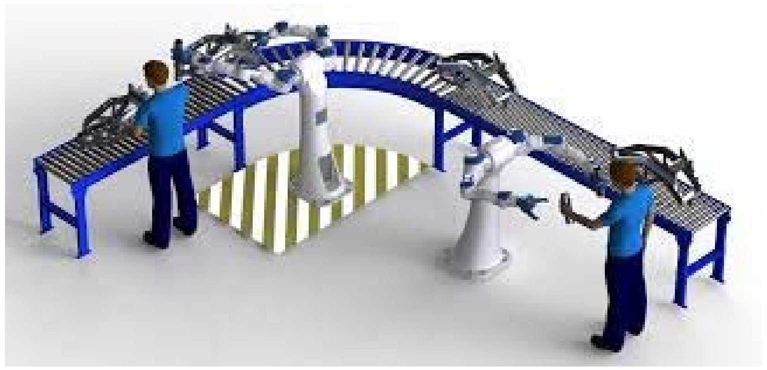
UPC

Institut d'Organització i Control de Sistemes Industrials

Outline

1. Introduction
1.1. Motivation

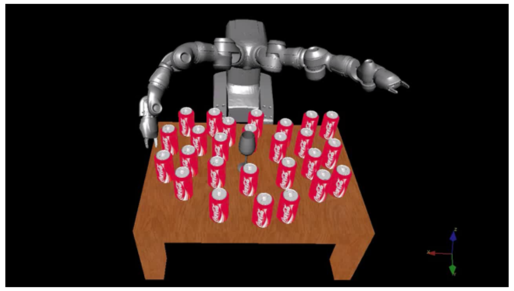
There are many interesting potential applications for dexterous and mobile collaborative robots, acting as robot helpers at home or as robot co-workers at the workplace.



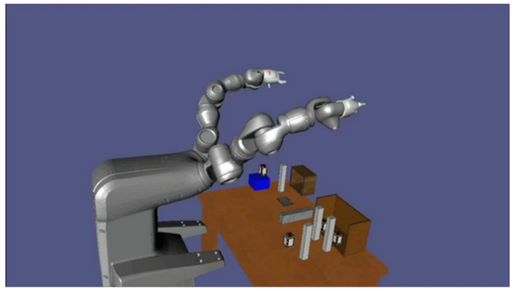
1. Introduction
1.2. Challenges and scope of the work done at IOC

Many challenges are related to robotic manipulation, which require robots to have **planning and execution capabilities**.

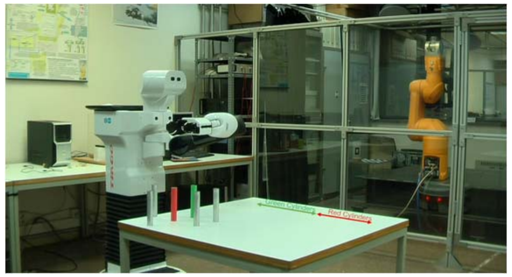
At **IOC** we have dealt with:



Physics-based motion planning capabilities to allow robot-objects interactions in cluttered manipulation environments.



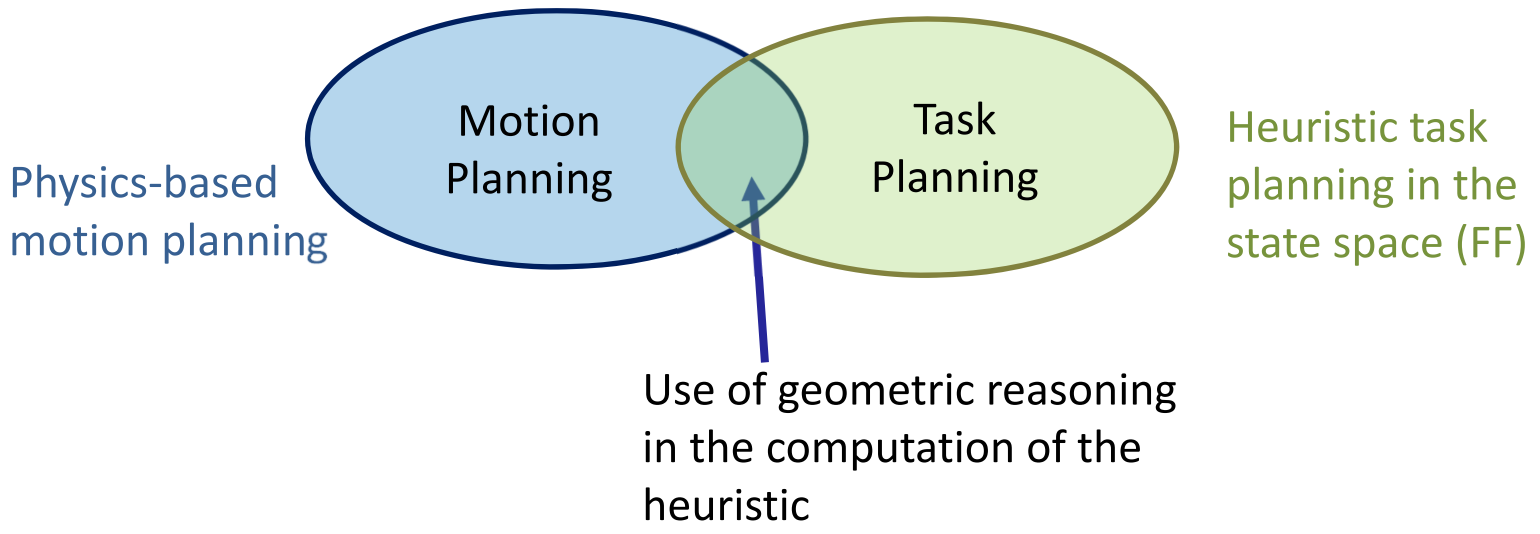
Combined Task and Motion Planning (TAMP) capabilities to cope with constrained manipulation problems.



Contingent TAMP capabilities to cope with manipulation tasks in semi-structured and uncertain environments.

1. Introduction
1.2. Challenges and scope of the work done at IOC

The focus of our research has been on:

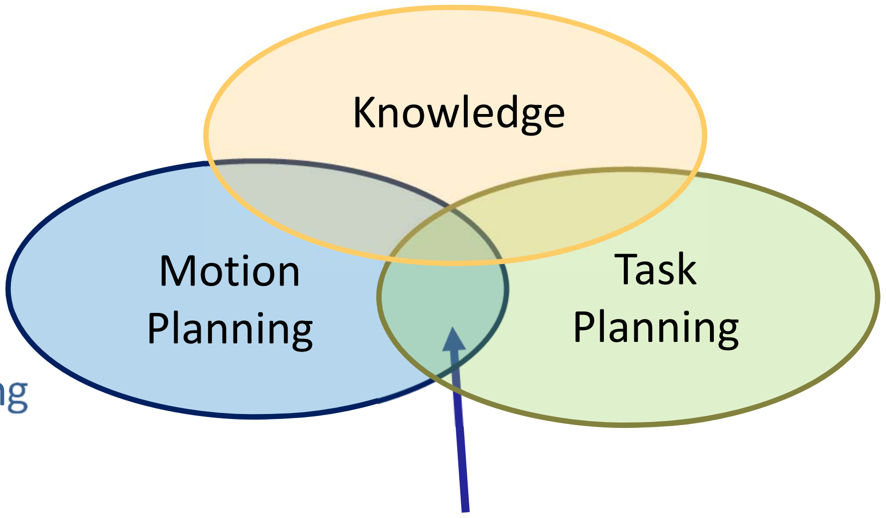

Motion Planning
Task Planning
Physics-based motion planning
Heuristic task planning in the state space (FF)
Use of geometric reasoning in the computation of the heuristic

2021 CASE IEEE Workshop on Smart Robotics Systems for Advanced Manufacturing Industries 7

1. Introduction
1.2. Challenges and scope of the work done at IOC

The focus of our research has been on:

Representation of knowledge using ontologies



Physics-based motion planning

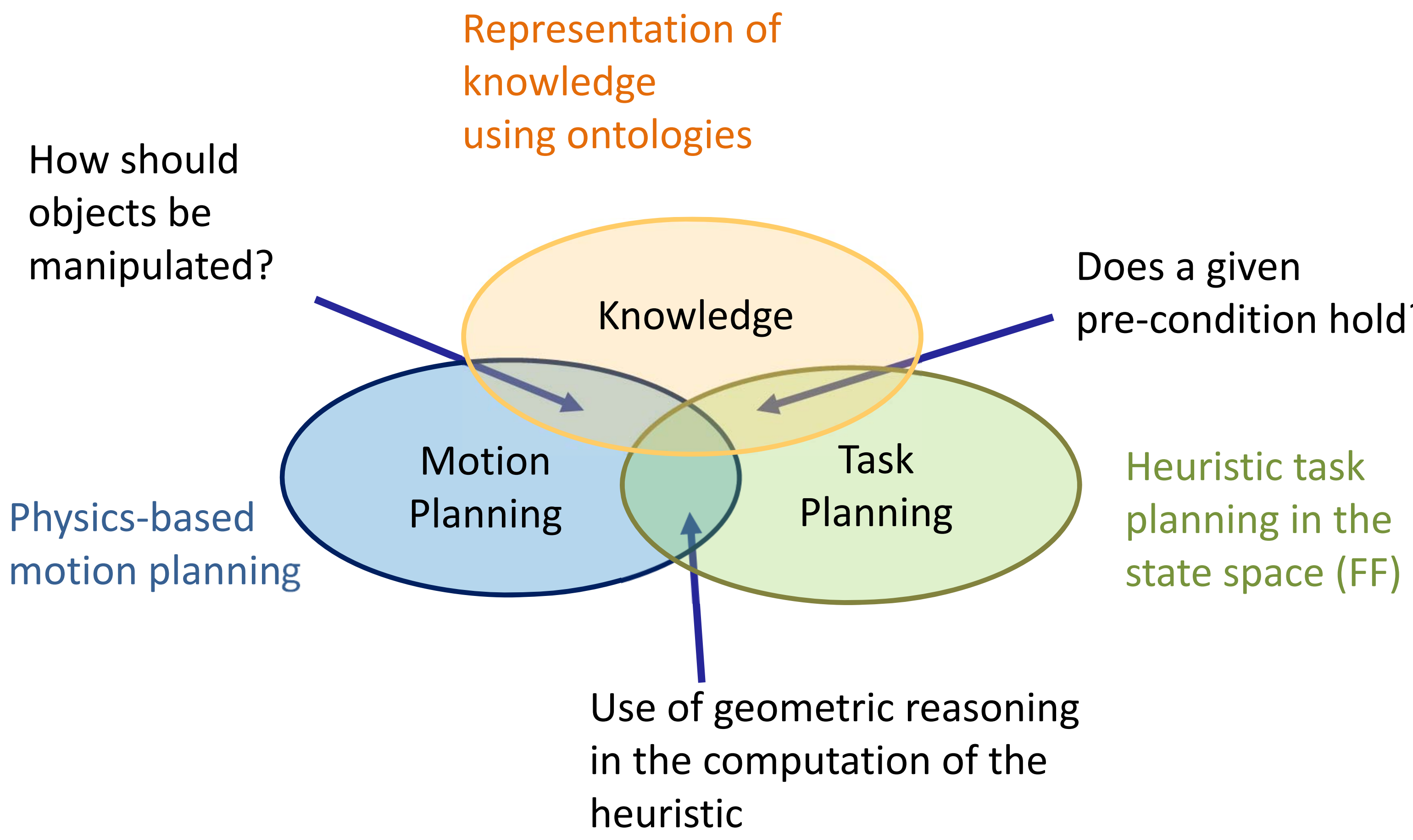
Heuristic task planning in the state space (FF)

Use of geometric reasoning in the computation of the heuristic

2021 CASE IEEE Workshop on Smart Robotics Systems for Advanced Manufacturing Industries 7

1. Introduction
1.2. Challenges and scope of the work done at IOC

The focus of our research has been on:


Representation of knowledge using ontologies
How should objects be manipulated?
Knowledge
Does a given pre-condition hold?
Motion Planning
Task Planning
Physics-based motion planning
Heuristic task planning in the state space (FF)
Use of geometric reasoning in the computation of the heuristic

2021 CASE IEEE Workshop on Smart Robotics Systems for Advanced Manufacturing Industries 7

1. Introduction
1.2. Challenges and scope of the work done at IOC

TAMP for robotic manipulation requires **reasoning** on:

Geometric information

- **Reachability reasoning** to evaluate whether an end-effector pose is reachable or not.
- **Spatial reasoning** to find a valid pose in a target placement region.
- **Manipulation reasoning** to evaluate feasible grasps for pick/place actions.

Actions

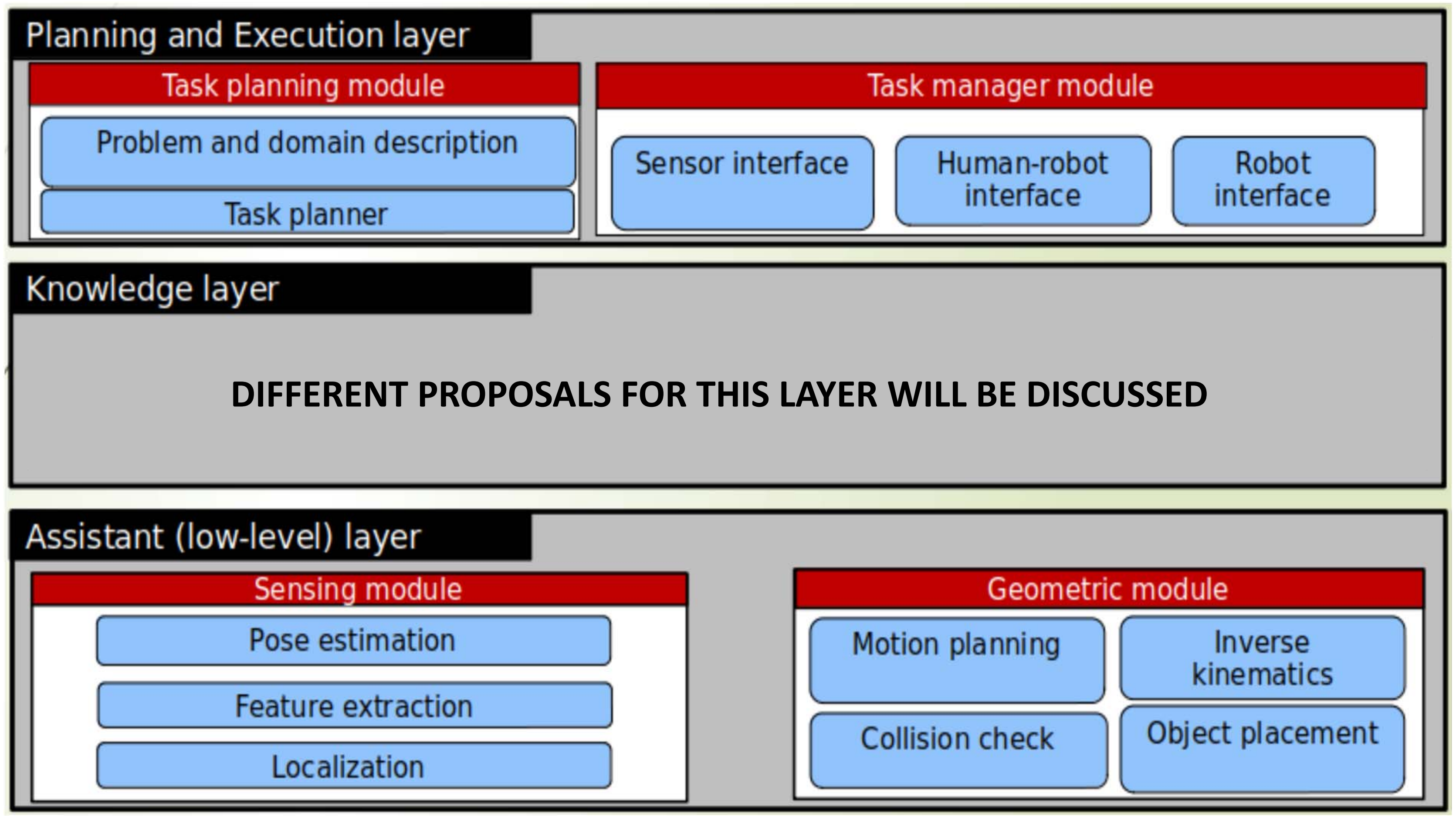
- **Reasoning on pre-conditions** to evaluate if they are satisfied for a given action
- **Reasoning on requirements** to evaluate which is the set of actions needed to solve a task
- **Reasoning for interaction** to guide motion planning
- **Reasoning on recovery strategies** to determine the actions required to recover from a failure

States

- **Reasoning on perception** to evaluate sensing requirements to evaluate the states
- **Reasoning on geometric constraints** to determine the current state
- **Reasoning on failures** to determine the occurrence of a failure
- **Reasoning on state similarity** to exploit previous experience

1. Introduction
1.2. Challenges and scope of the work done at IOC

The basic framework is:



1. Introduction
1.2. Challenges and scope of the work done at IOC

1. Table-top manipulation problems with focus on:

- geometric reasoning for the integration of task and motion planning levels,
- perception,
- action feasibility (pick/push)
- interaction for motion planning.

- **Knowledge-oriented for Task and Motion Planning** (KTAMP) framework uses the Perception and Manipulation Knowledge (PMK) ontology to integrate task and motion planning.

2. Assembly manipulation problems with focus on:

- state reasoning,
- action pre-conditions and results,
- failure detection and recovery.

- **Failure Interpretation and Recovery Ontology** (FailRecOnt) framework offers a flexible reasoning tool that is perfectly adapted to knowledge-driven planning schemes

1. Introduction
1.2. Challenges and scope of the work done at IOC

3. Mobile-based every-day manipulation problems with focus on:

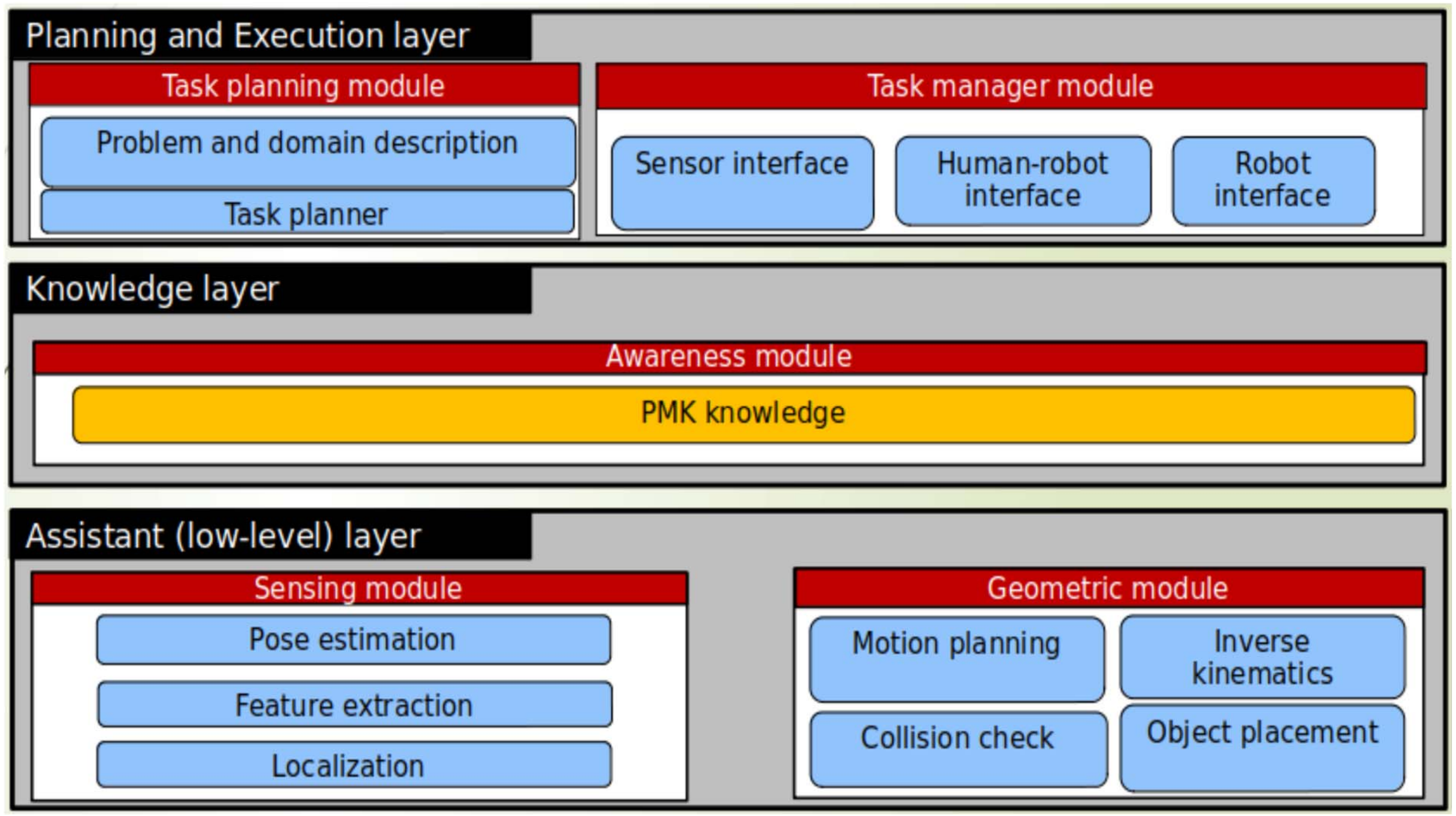
- state similarity (to efficiently re-use previous known ways to execute actions),
- task requirements,
- robustness.

- A **Skill-based Robotic Manipulation Framework** (SkillMaN) integrates the previous tools with a similarity situation evaluator and a motion adaptation tool.
- A **robust behaviour tree-based execution framework** that flexibly adapts to the task requirements and to the current (possible unexpected) states.

Outline

2. Reasoning on manipulation actions
2.1. Framework

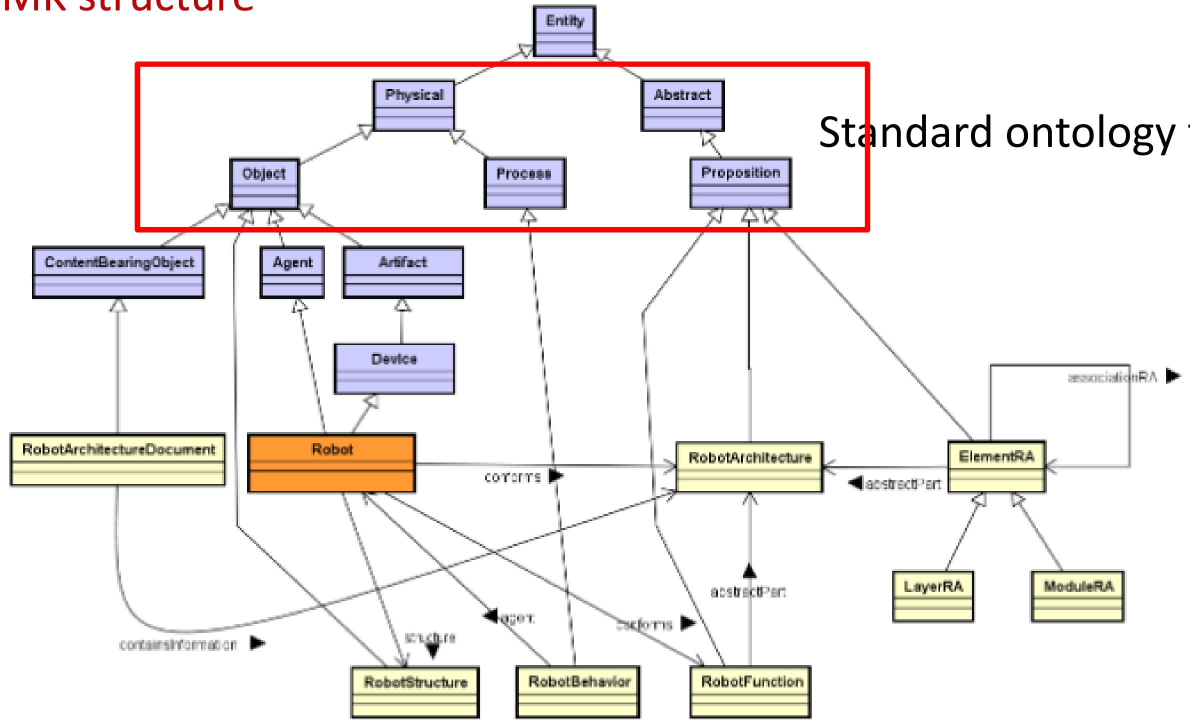
PMK: Perception and Manipulation Knowledge ontology



2. Reasoning on manipulation actions
2.1. Framework

PMK: Perception and Manipulation Knowledge ontology

Modeling: PMK structure



Standard ontology frameworks

Fig. 1: Overview of the ontology's taxonomy and relations depicted in standard UML class diagram notation. Blue boxes are concepts from SUMO. Orange boxes are concepts from CORA. Yellow boxes are concepts from ROA Ontology. Almost all relations are imported from SUMO/CORA, with exception of *structure* and *associationRA*.

(Howard et al., 2015)

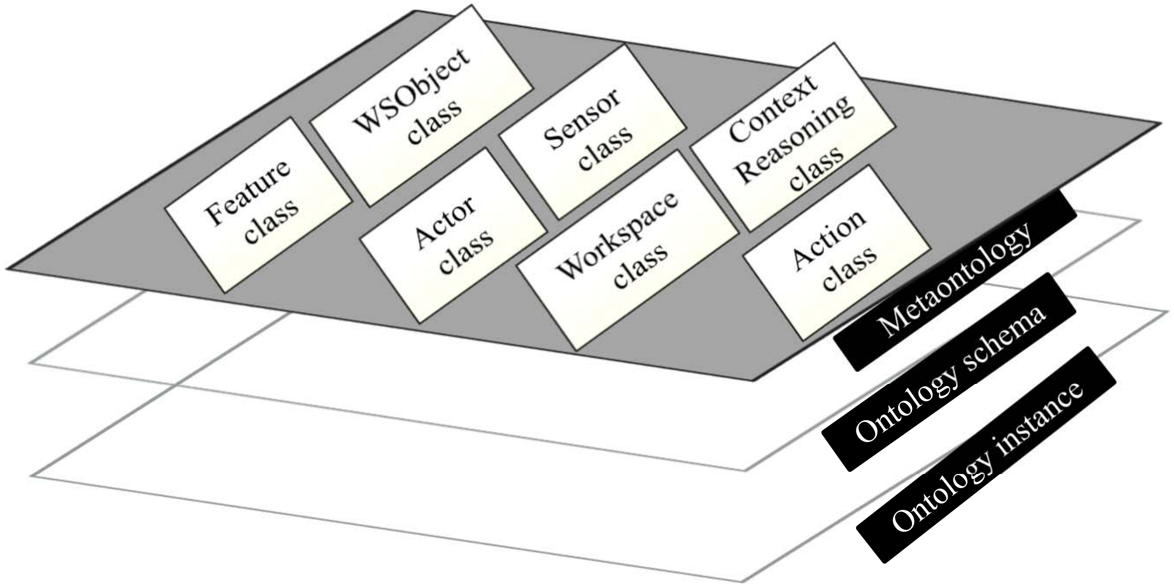
2. Reasoning on manipulation actions
2.1. Framework

PMK: Perception and Manipulation Knowledge ontology

Modeling: PMK structure

Layers:
Metaontology
Ontology schema
Ontology instance

Classes:
1. Feature
2. WSObject
3. Actor
4. Sensor
5. Workspace
6. Context Reasoning
7. Action

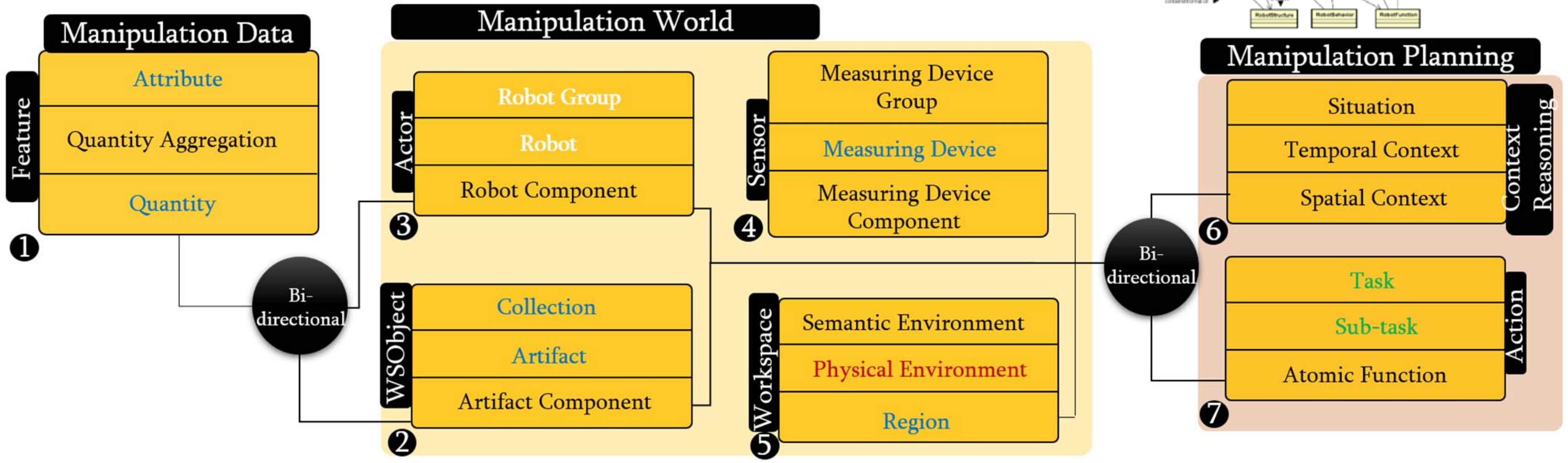


2. Reasoning on manipulation actions
2.1. Framework

PMK: Perception and Manipulation Knowledge ontology

Modeling: PMK structure

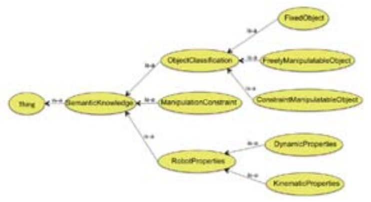
Entity
Physical
Abstract
Object
Process
Proposition
ContentBearingObject
Agent
Artifact
Device
RobotArchitectureDocument
Robot
RobotArchitecture
ElementRA
LayerRA
ModuleRA
RobotStructure
RobotBehavior
RobotFunction



Structure of the metaontology layer of PMK fit to follow the standard IEEE-1872. Concepts introduced are shown in black while those that inherit from the standard are shown in colors: SUMO (blue), CORA (white), CORAX (red), and ROA (green).

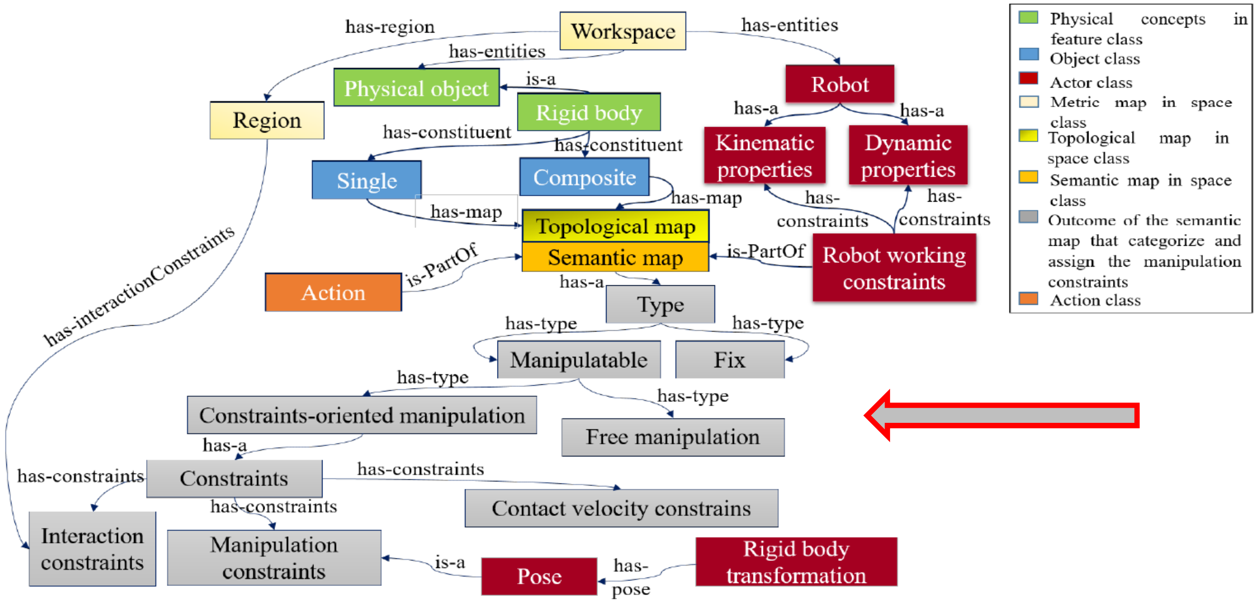
Reasoning for physics-based motion planning

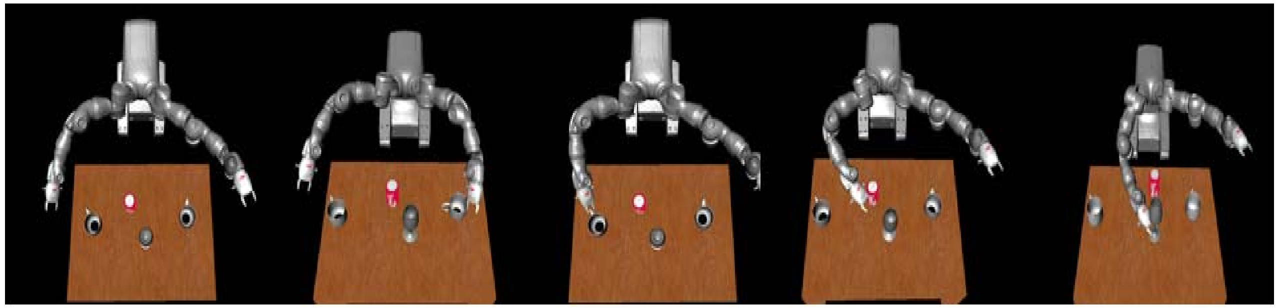
PMK Knowledge



Prolog Inference Process

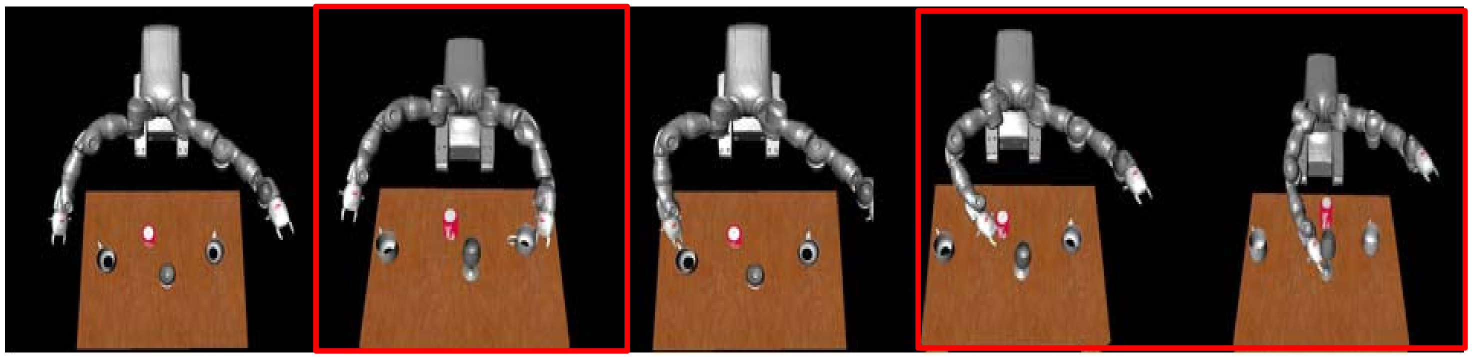
Motion Planner





Reasoning for physics-based motion planning

Example: Query to the knowledge on how to push a cup.



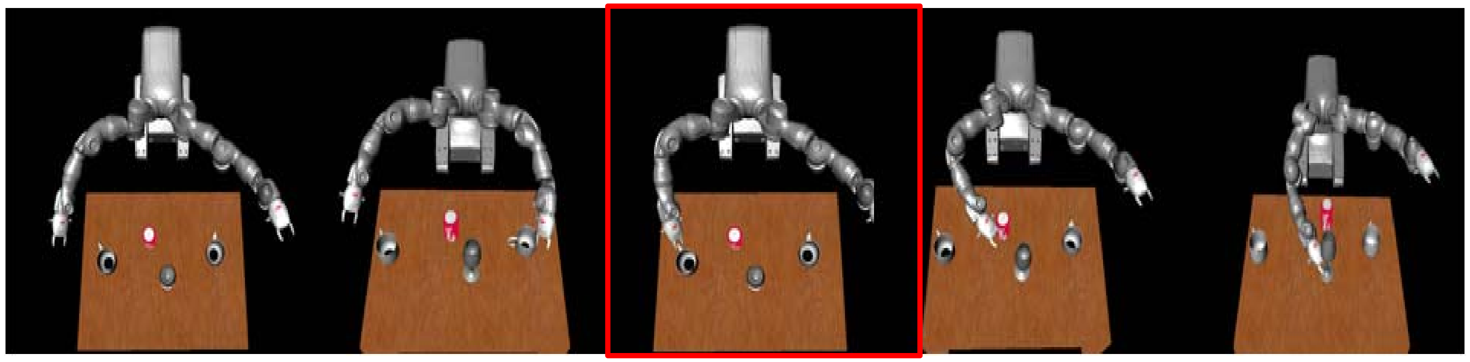
Question 1: This question describes *"how to push a rigid body?"*

```
?— pushRigidBody( hasAction(action , object) ).
Answer: Pushable object = object.
```

$$
\begin{aligned}
&Cup := rigidbody\\
&\wedge\exists hasSuperclass(PhysicalObject, Rigidbody)\\
&\wedge\exists hasPart(Cup, handle)\\
&\wedge\exists hasPart(Cup, body)\\
&\boxed{\wedge\exists hasAction(Cup, push)}\\
&\wedge\exists hasType(Cup, manipulatable)\\
&\boxed{\wedge\exists hasInteractionRegion(Cup, body)}\\
&\wedge\neg\exists hasInteractionRegion(Cup, handle)\\
&\wedge\exists hasInteractionVelocity(body, velocity)\\
&\wedge\exists hasInteractionParameter(body, physicalProperties)\\
&\wedge\exists hasInteractionParameter(gripper, physicalProperties)
\end{aligned}
$$

Reasoning for motion planning for grasp actions

Example: Query to the knowledge on how to pick a cup.



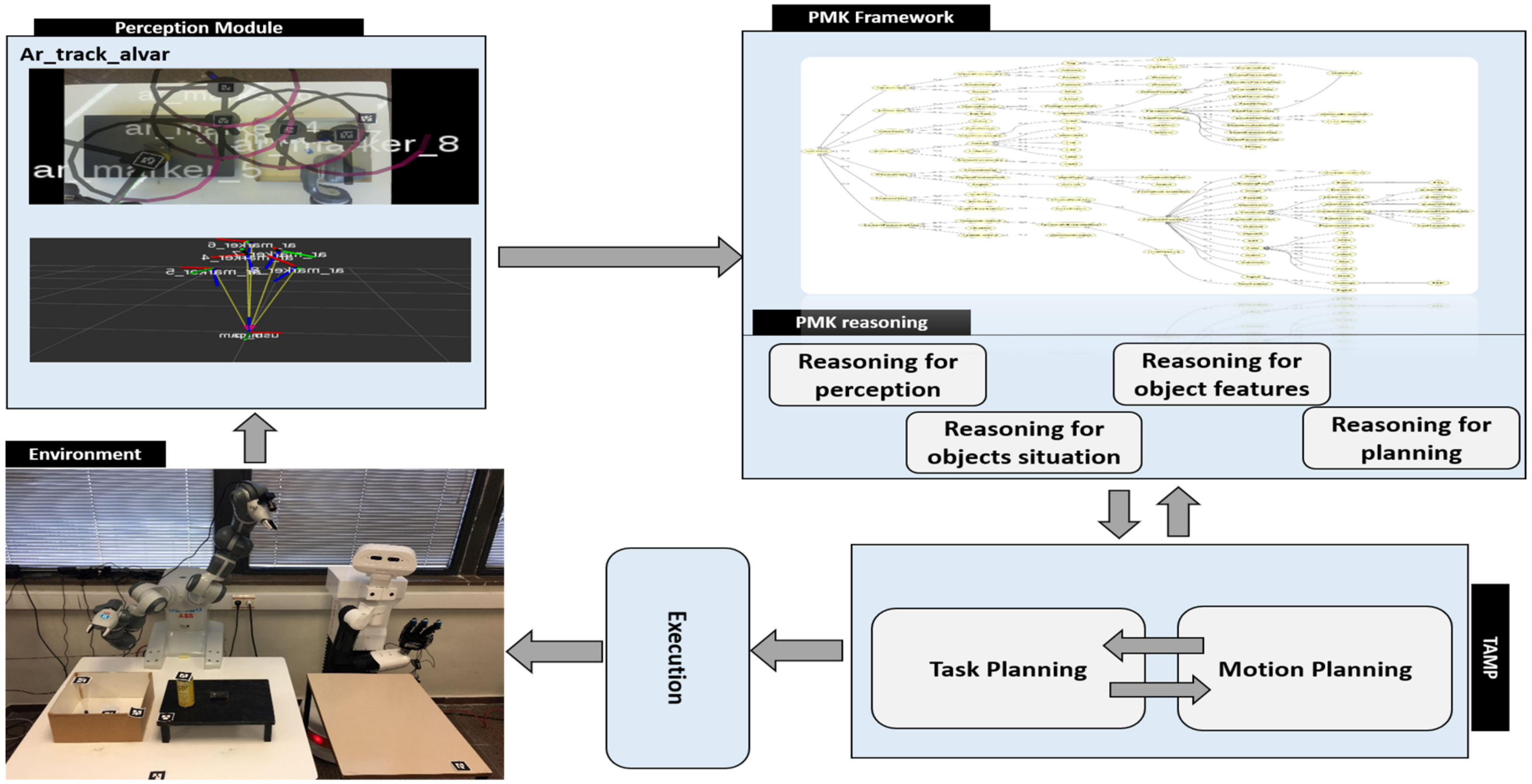
Question 2: This question describes *"how to pick a rigid body?"*

```
?– pick( hasAction(action , object) ).
Answer: pickObject = object.
```

$Cup := rigidbody$
$\wedge \exists hasSuperclass(PhysicalObject, Rigidbody)$
$\wedge \exists hasPart(Cup, handle)$
$\wedge \exists hasPart(Cup, body)$
$\boxed{\wedge \exists hasAction(Cup, pick)}$
$\wedge \exists hasType(Cup, manipulatable)$
$\boxed{\wedge \exists hasInteractionRegion(Cup, handle)}$
$\wedge \neg \exists hasInteractionRegion(Cup, body)$
$\wedge \exists hasInteractionVelocity(body, velocity)$
$\wedge \exists hasInteractionParameter(handle, physicalProperties)$
$\wedge \exists hasInteractionParameter(gripper, physicalProperties)$
$\wedge \exists hasGraspingPose(cup, graspingPose)$

2. Reasoning on manipulation actions
2.3. Reasoning for task planning

KTAMP = TAMP + PMK



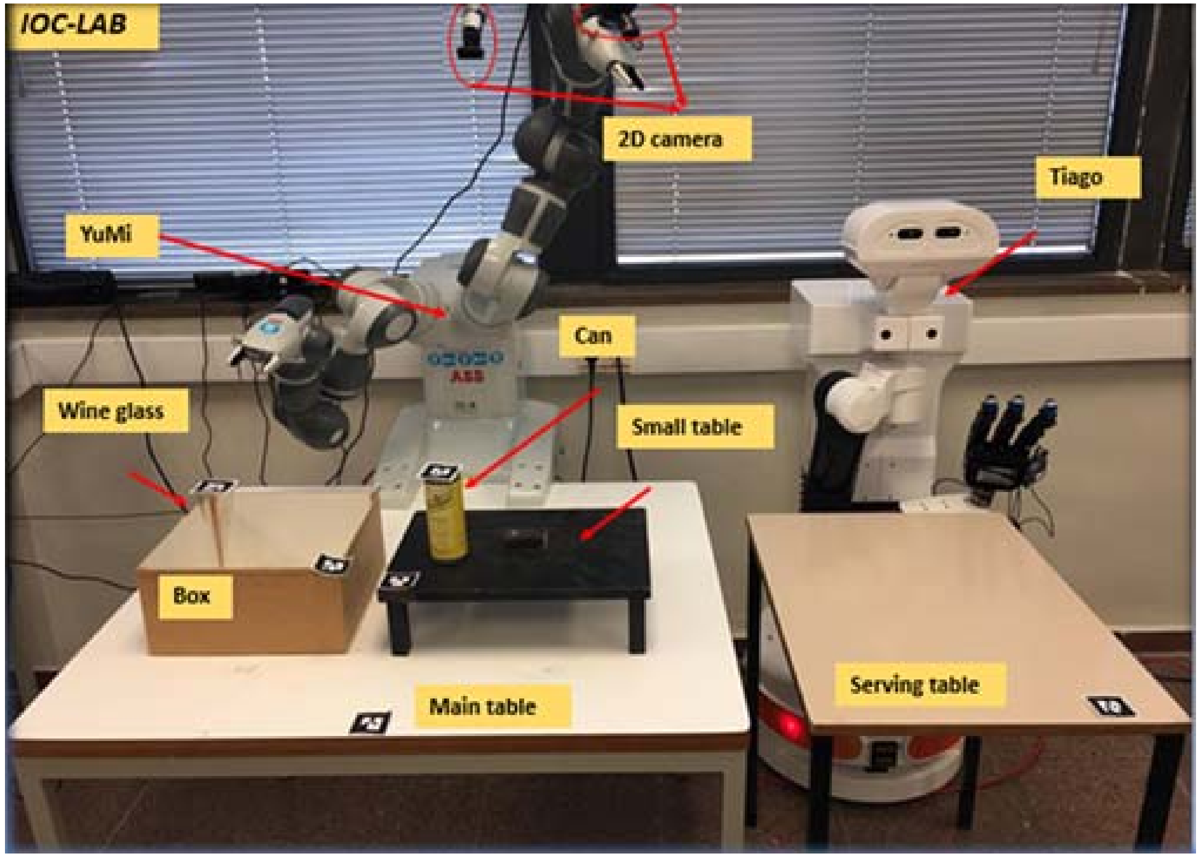
Main parts of the system: The perception module, PMK framework and TAMP planning module.

PMK asserts the perceptual data, builds the IOC-Lab knowledge, and provides the reasoning predicates to the planning module.

2. Reasoning on manipulation actions
2.3. Reasoning for task planning

KTAMP = TAMP + PMK

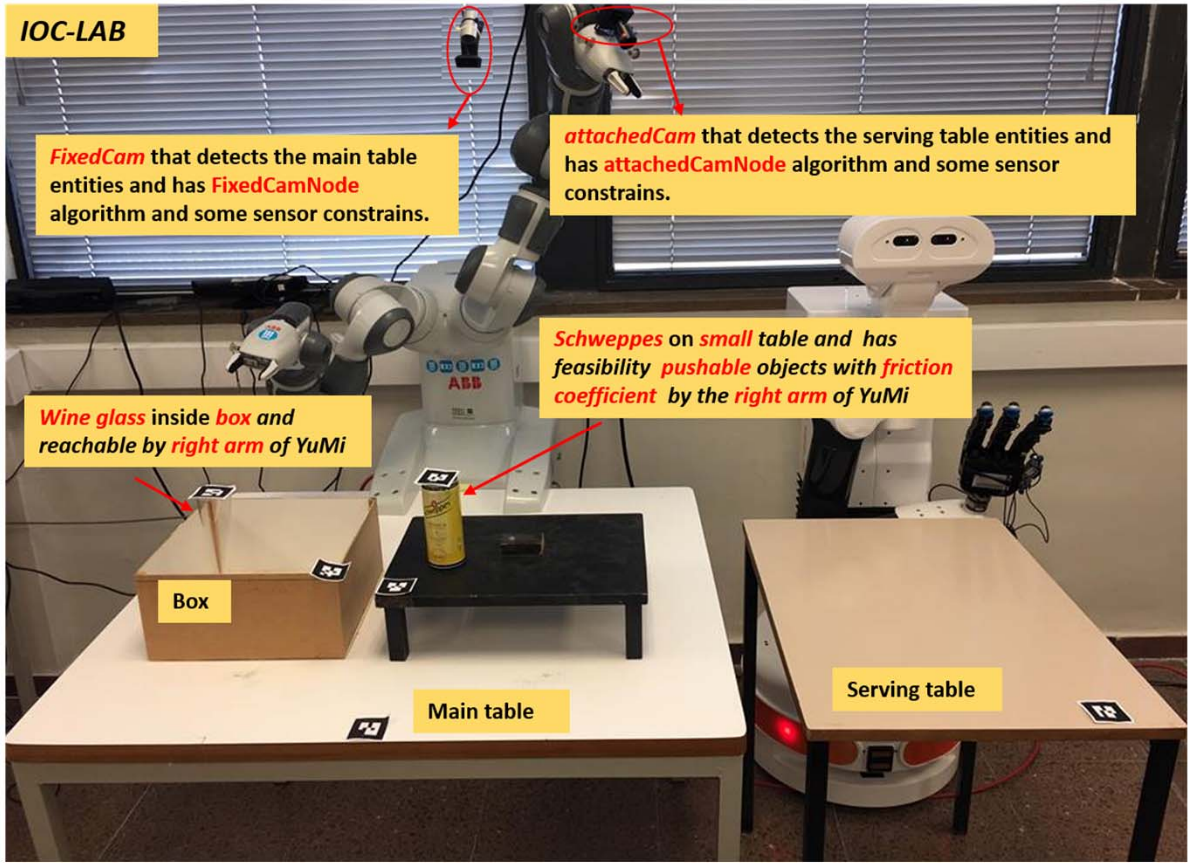
Scenario: a two-robot table-top manipulation task at the IOC-Lab



2. Reasoning on manipulation actions
2.3. Reasoning for task planning

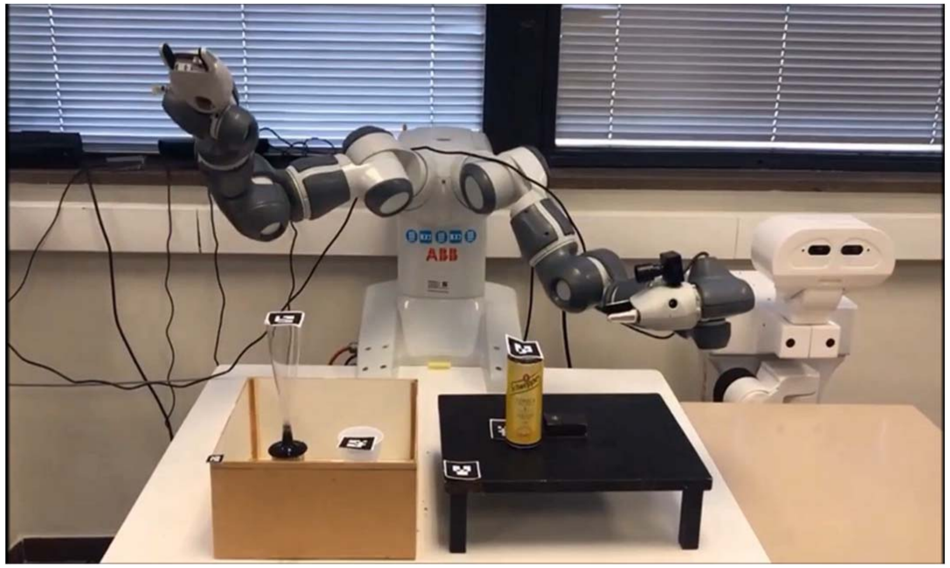
KTAMP = TAMP + PMK

Scenario: a two-robot table-top manipulation task at the IOC-Lab


IOC-LAB
FixedCam that detects the main table entities and has FixedCamNode algorithm and some sensor constrains.
attachedCam that detects the serving table entities and has attachedCamNode algorithm and some sensor constrains.
Schweppes on small table and has feasibility pushable objects with friction coefficient by the right arm of YuMi
Wine glass inside box and reachable by right arm of YuMi
Box
Main table
Serving table

2. Reasoning on manipulation actions
2.3. Reasoning for task planning

KTAMP = TAMP + PMK



2. Reasoning on manipulation actions
2.3. Reasoning for task planning

KTAMP = TAMP + PMK

$$
\begin{aligned}
&\boxed{RunFixedNode :-}\\
&\exists hasSuperclass(subTask, Action)\\
&\wedge \exists Is\text{-}a(FindObject, subTask)\\
&\wedge \exists Is\text{-}a(FixedCam, measuringDevice)\\
&\wedge \exists hasfeatureOfInterset(2DCamera, Image)\\
&\wedge \exists hasObseredProperty(Image, location)\\
&\wedge \exists hasObseredProperty(Image, ID)\\
&\wedge \exists useProcess(FixedCam, FixedNode)
\end{aligned}
$$

$$
\begin{aligned}
&\boxed{ManipulationConstraints :-}\\
&\exists hasSuperclass(artifact, WSobject)\\
&\wedge \exists hasmaterial(material, artifact)\\
&\wedge \exists hasmass(mass, artifact)\\
&\wedge \exists hasInterParameter(interactParameter, artifact)\\
&\wedge \exists hasManReg(manipulatableRegion, artifact)
\end{aligned}
$$

$$
\begin{aligned}
&\boxed{OccupiedPlacementRegion :-}\\
&\exists hasregionID(obj, ID)\\
&\wedge \exists hasspatialRelation(obj, on).
\end{aligned}
$$

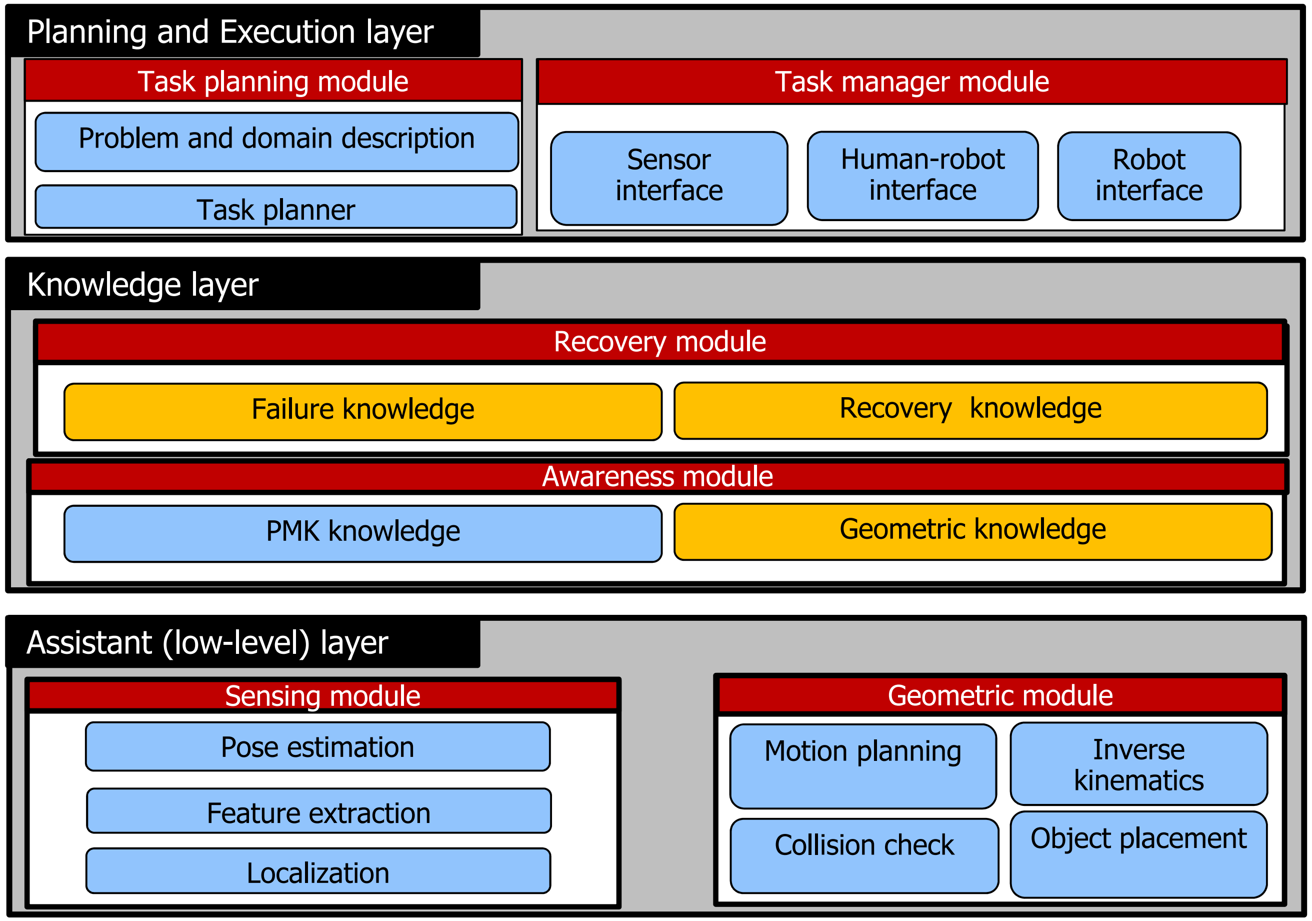
$$
\begin{aligned}
&\boxed{Pickup :-}\\
&\exists hasSuperclass(Pick, Action)\\
&\wedge \forall hasTaskTarget(Pick, Artifact)\\
&\wedge \exists hasArm(Robot, Arm)\\
&\wedge \exists hasConstrains(Artifact, Top)\\
&\wedge \exists hasGaspingPose(Artifact, GraspingPose)\\
&\wedge \exists hasObjectPose(Artifact, ObjectPose)\\
&\wedge \exists hasdimension(Artifact, Bbox_1)\\
&\wedge \exists isMemberOf(Gripper, Robot)\\
&\wedge \exists hasdimension(Gripper, Bbox_2)\\
&\wedge \exists fitInside(Bbox_1, Bbox_2)\\
&\wedge \exists hasCapability(Payload, Gripper)
\end{aligned}
$$

Outline

3. Heterogeneous reasoning

3.1. Framework

FailRecOnt: Failure Detection and Recovery ontology



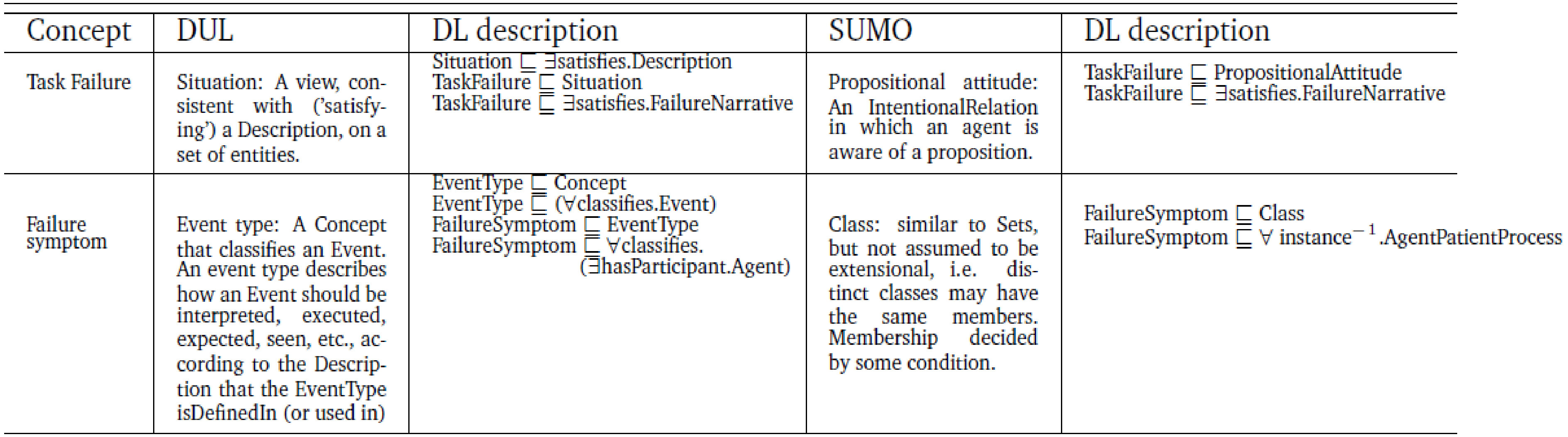
3. Heterogeneous reasoning
3.1. Framework

FailRecOnt: Failure Detection and Recovery ontology

Modeling the ontologies in different foundations

SUMO provides a conceptual structure that can be used and integrated with other specific ontologies developed for the robotics and automation domain

DUL makes a variety of important distinctions that are useful for upper ontologies. Moreover, axiomatization makes use of simple constructions



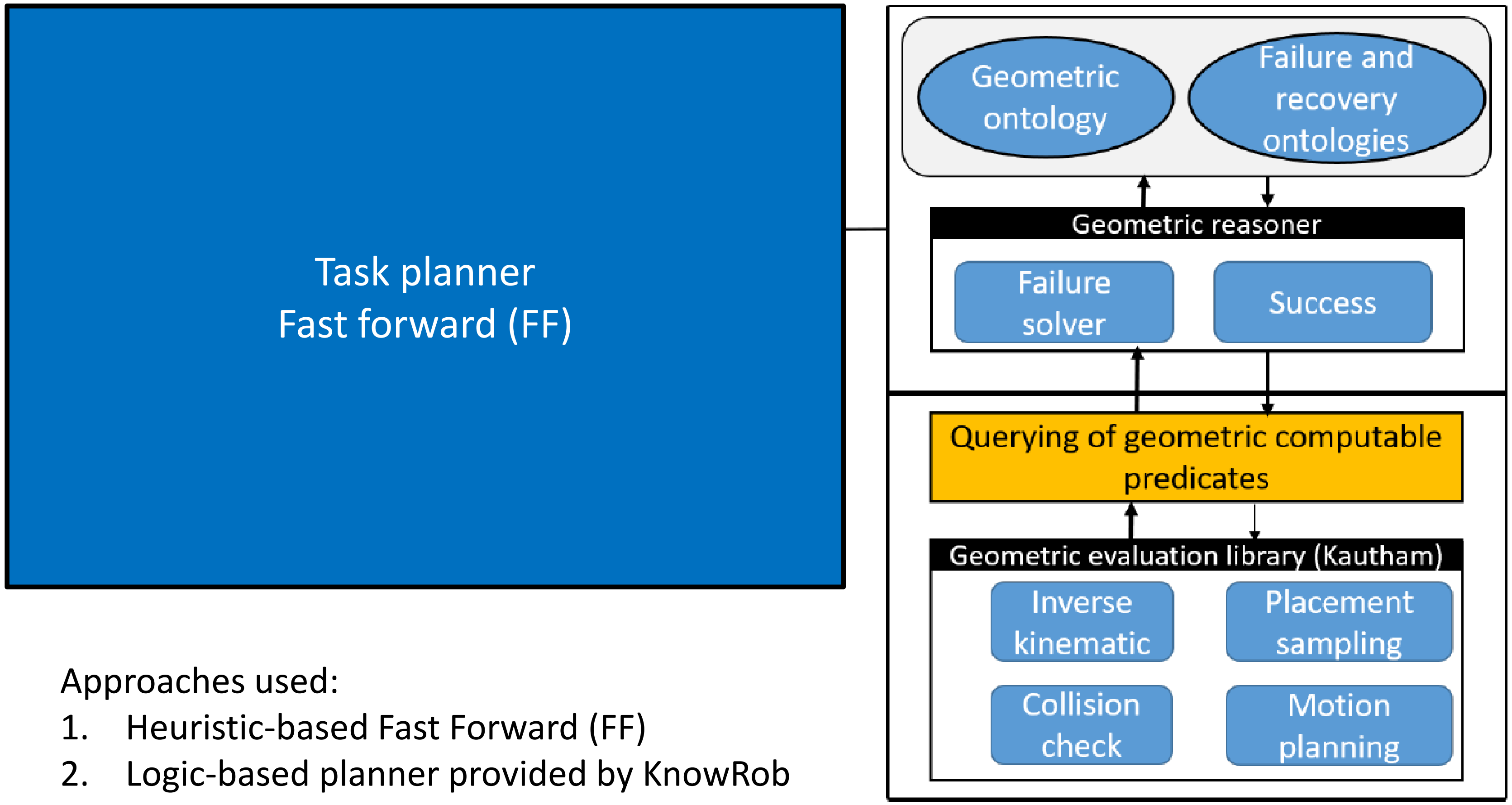
Concept	DUL	DL description	SUMO	DL description
Task Failure	Situation: A view, consistent with ('satisfying') a Description, on a set of entities.	Situation $\sqsubseteq$ $\exists$satisfies.Description TaskFailure $\sqsubseteq$ Situation TaskFailure $\sqsubseteq$ $\exists$satisfies.FailureNarrative	Propositional attitude: An IntentionalRelation in which an agent is aware of a proposition.	TaskFailure $\sqsubseteq$ PropositionalAttitude TaskFailure $\sqsubseteq$ $\exists$satisfies.FailureNarrative
Failure symptom	Event type: A Concept that classifies an Event. An event type describes how an Event should be interpreted, executed, expected, seen, etc., according to the Description that the EventType isDefinedIn (or used in)	EventType $\sqsubseteq$ Concept EventType $\sqsubseteq$ ($\forall$classifies.Event) FailureSymptom $\sqsubseteq$ EventType FailureSymptom $\sqsubseteq$ $\forall$classifies.($\exists$hasParticipant.Agent)	Class: similar to Sets, but not assumed to be extensional, i.e. distinct classes may have the same members. Membership decided by some condition.	FailureSymptom $\sqsubseteq$ Class FailureSymptom $\sqsubseteq$ $\forall$ instance^{-1}.AgentPatientProcess

3. Heterogeneous reasoning

3.1. Framework

FailRecOnt: Failure Detection and Recovery ontology

Integration of geometric reasoning module within ontology



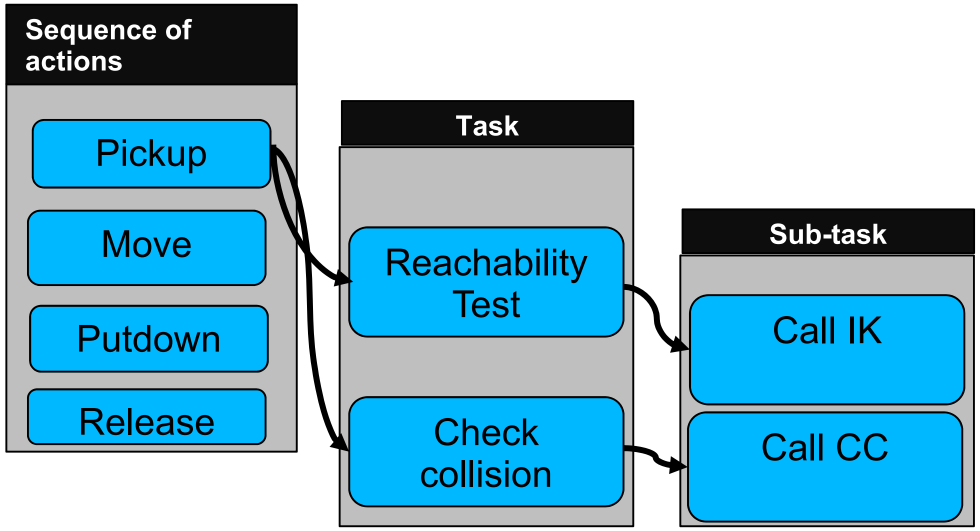
Approaches used:

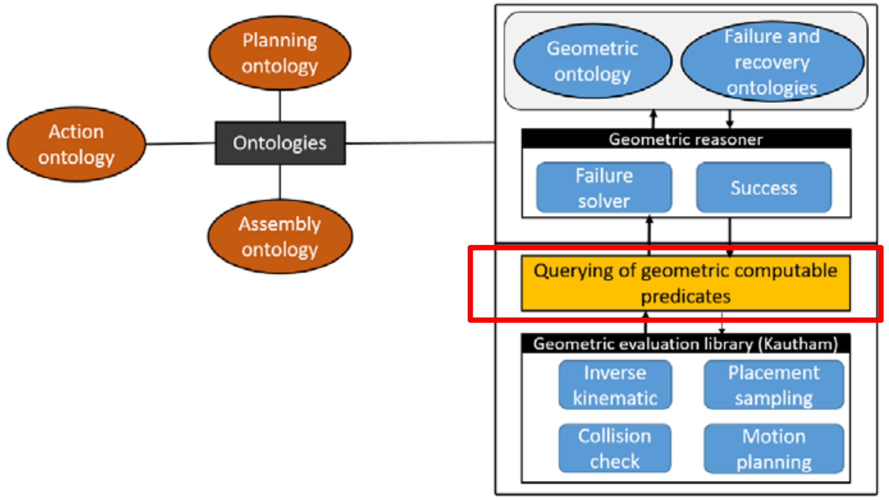
1. Heuristic-based Fast Forward (FF)
2. Logic-based planner provided by KnowRob

3. Heterogeneous reasoning
3.1. Framework

FailRecOnt: Failure Detection and Recovery ontology

Integration of geometric reasoning module within ontology





"Is the path toward a goal configurations reachable?":

```
?– testReachability( hasAlgorithm(IKModule, IKAlgorithm) ).
Algorithm = Call IK service.
```

"Is the path toward a goal configurations collision-free?":

```
?– CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ).
Algorithm = Call collision check service.
```

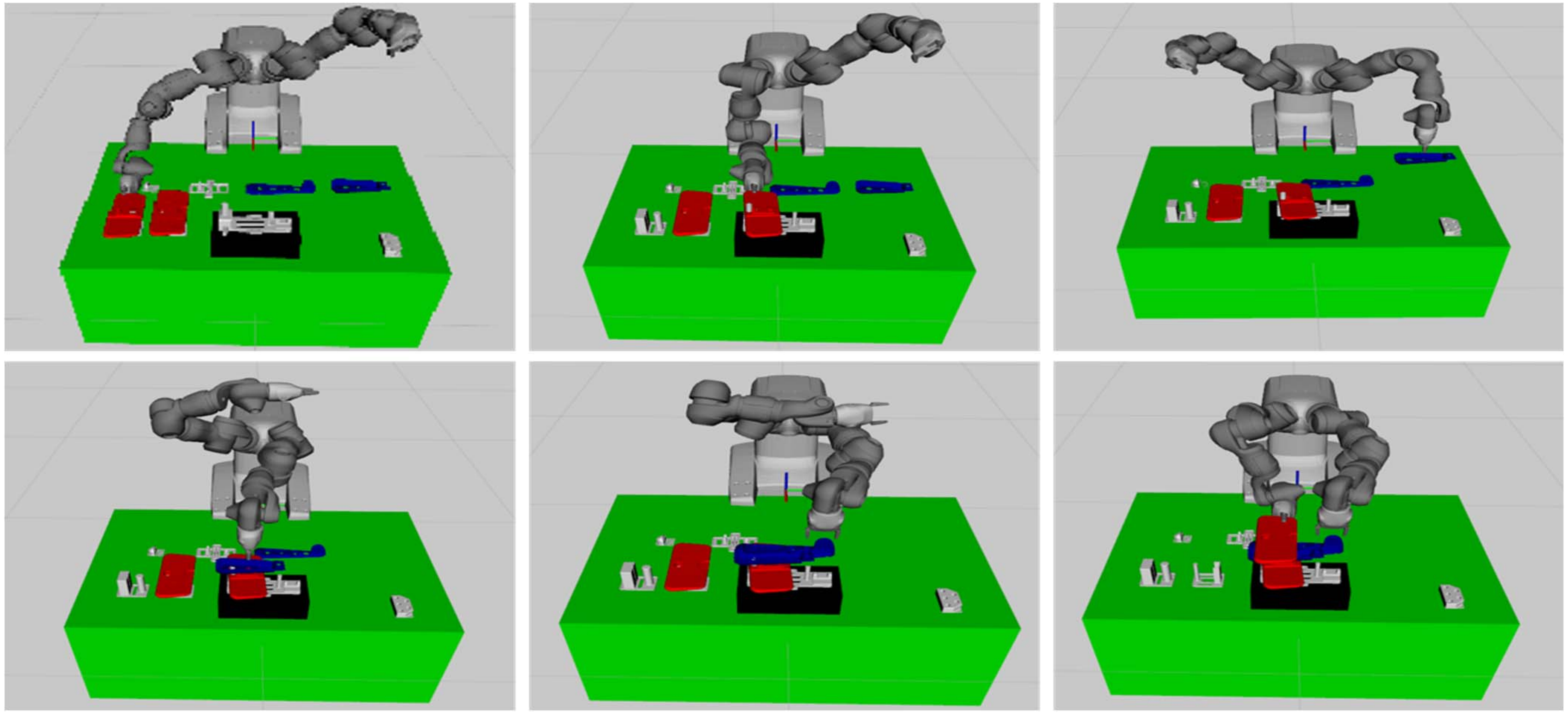
```
?– testReachability( hasAlgorithm(IKModule, IKAlgorithm) ),
?– CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ),
?– MotionPlanning( hasAlgorithm(MPModule, MPAlgorithm) ).
```

```
Algorithm = Call [IK, CC, MP].
```

3. Heterogeneous reasoning
3.1. Framework

FailRecOnt: Failure Detection and Recovery ontology

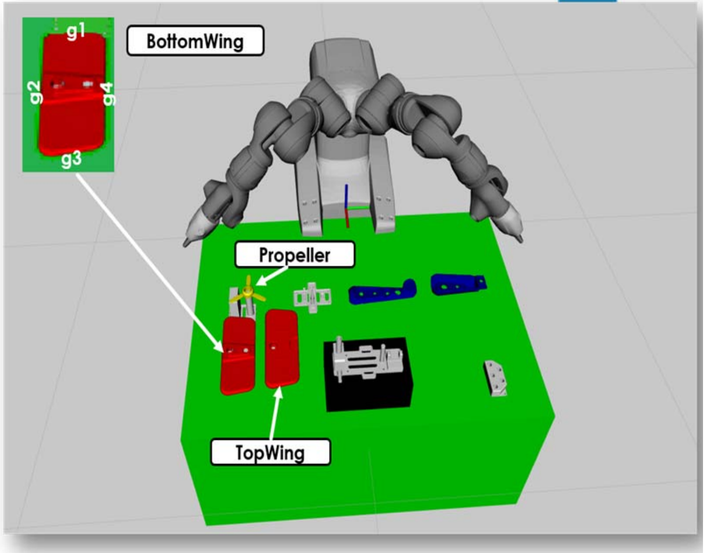
Test scenario: Battat plane assembly



3. Heterogeneous reasoning
3.2. Geometric reasoning

What are the occluded parts required in a connection?

```
?- holds( needsAffordance(Connection,Affordance) ),
   holds( hasAffordance(Occluded,Affordance) ),
   holds( partOccludedBy(Occluded,OccludingPart) ).
Occluded='PlaneBottomWing1', OccludingPart='PlaneUpperBody1'.
```



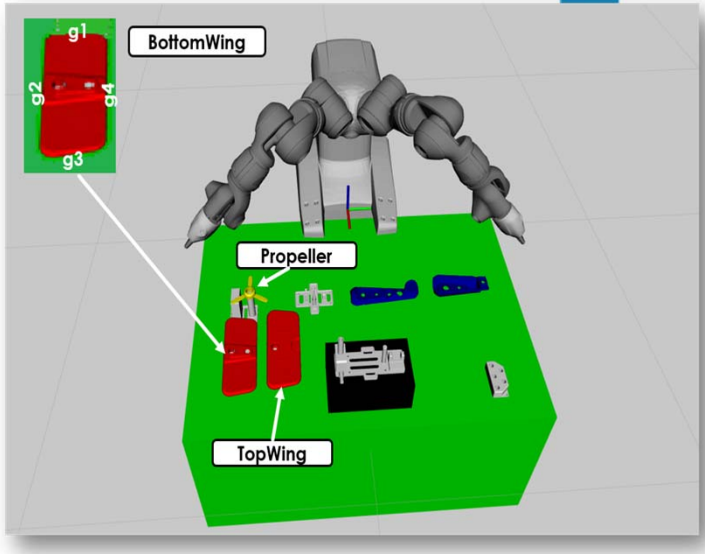
What action pre-conditions are not fulfilled and what can be done do to fix this?

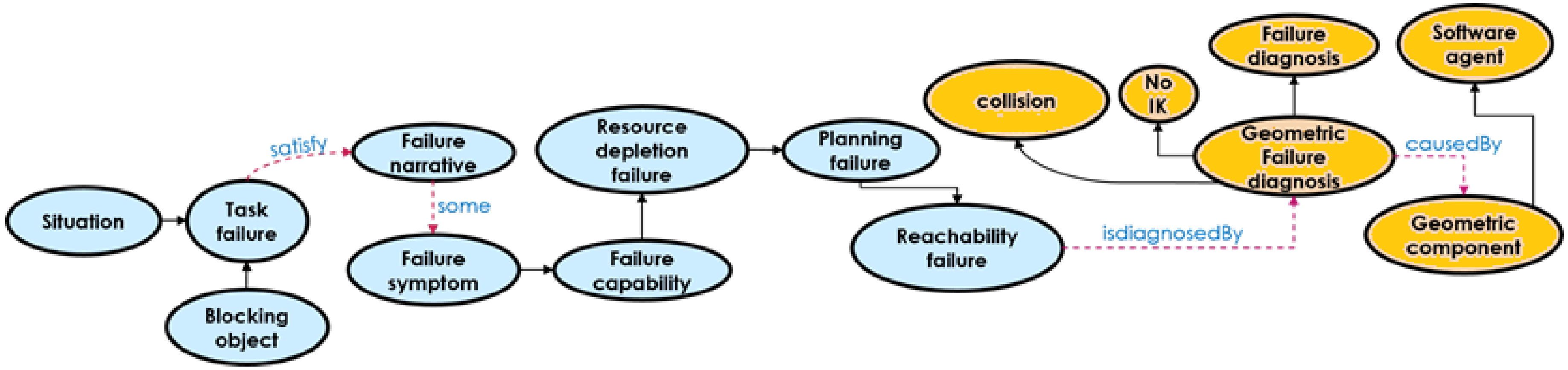
```
?- entity(Act, [an,action, [type, 'ConnectingParts'],
       [assemblesConnection, Connection]]),
   agenda_create(Act, Agenda),
   agenda_next_item(Agenda,Item).
 Item = "detach PlaneBottomWing1 partOccludedBy PlaneUpperBody1"
```

What action it should be performed to dissolve the *partOccludedBy* relation between parts

```
?- holds( usesActionMapper(Strategy,Mapper) ),
   property_range(Mapper,mapsItem,Pattern),
   individual_of(Item,Pattern),
   call(Mapper,Item,Action).
 Action = [an,action, [type, 'PutAwayPart'],
       [movesPart, 'PlaneUpperBody1'], ...].
```

3. Heterogeneous reasoning
3.3. Reasoning on failures

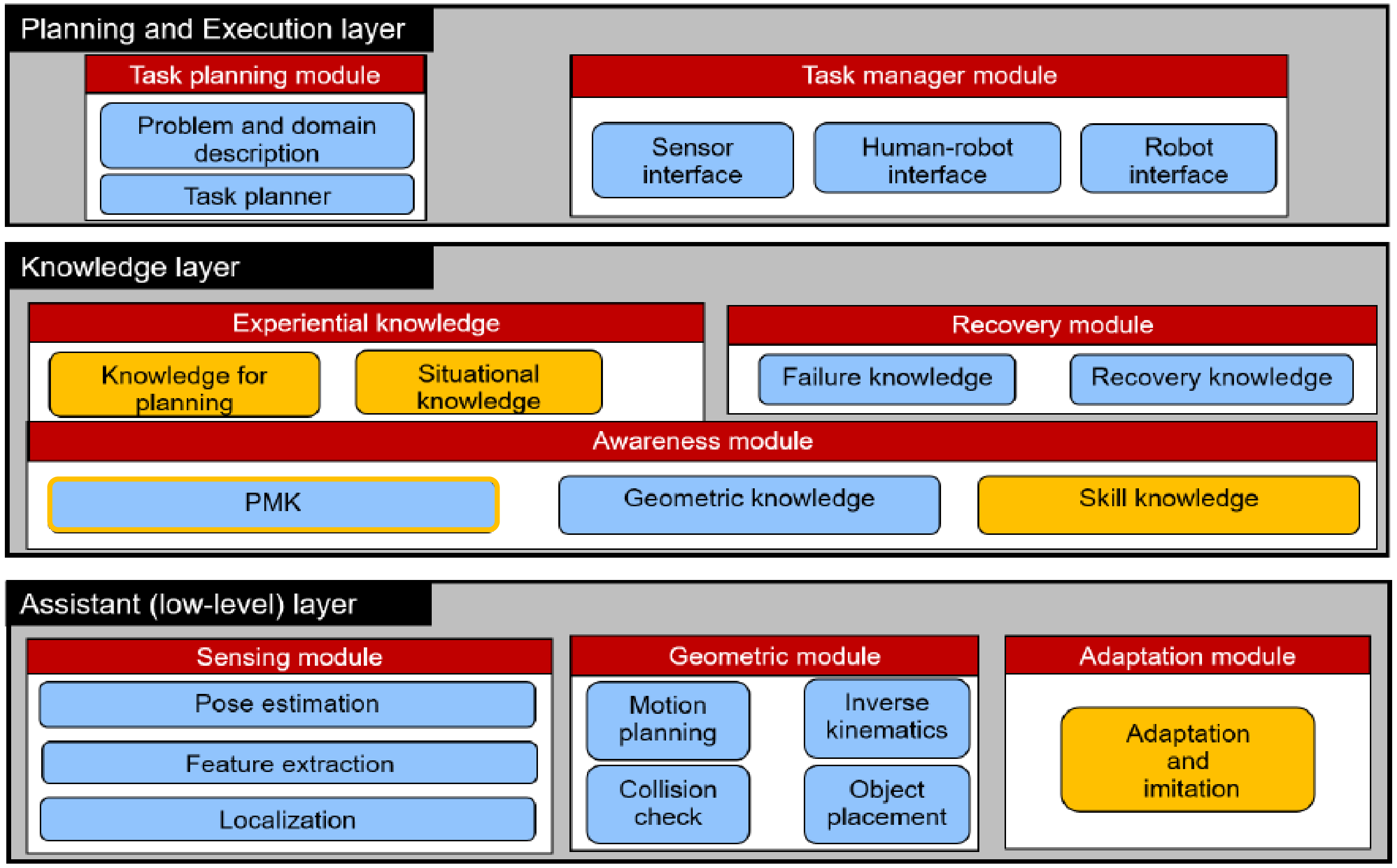




Outline

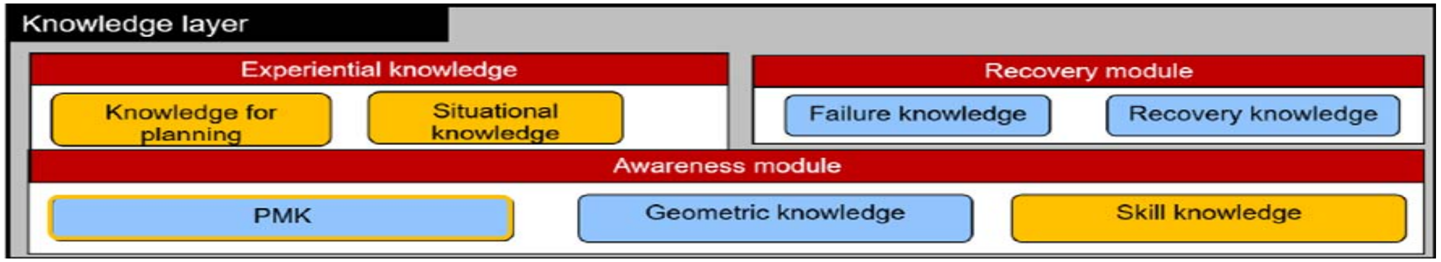
4. Reasoning for adaptation
4.1. Framework

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning



4. Reasoning for adaptation
4.1. Framework

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning

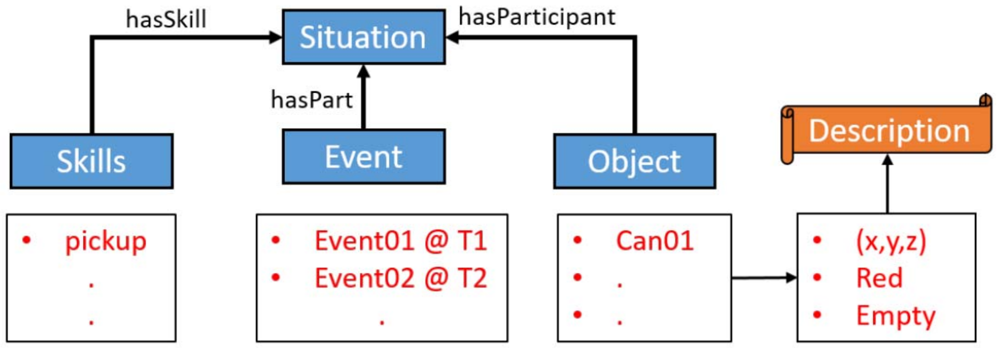


Knowledge for planning

Provides the geometric-skills information (based on the robot's experience, such as how to grasp an object?

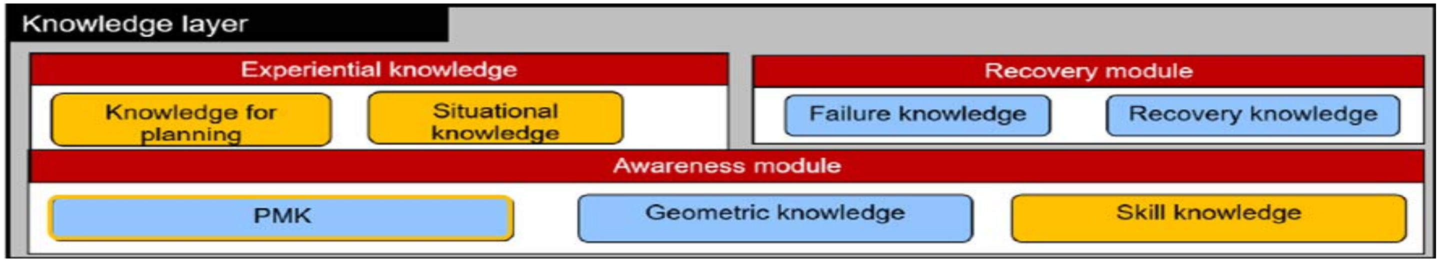
Situational knowledge

Provides the similarity check between the current situations with those stored in a database



4. Reasoning for adaptation
4.1. Framework

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning



Skill knowledge

Provides the available skills in the knowledge database to be used by the planning module, and how to execute them

Assistant layer: adaptation module

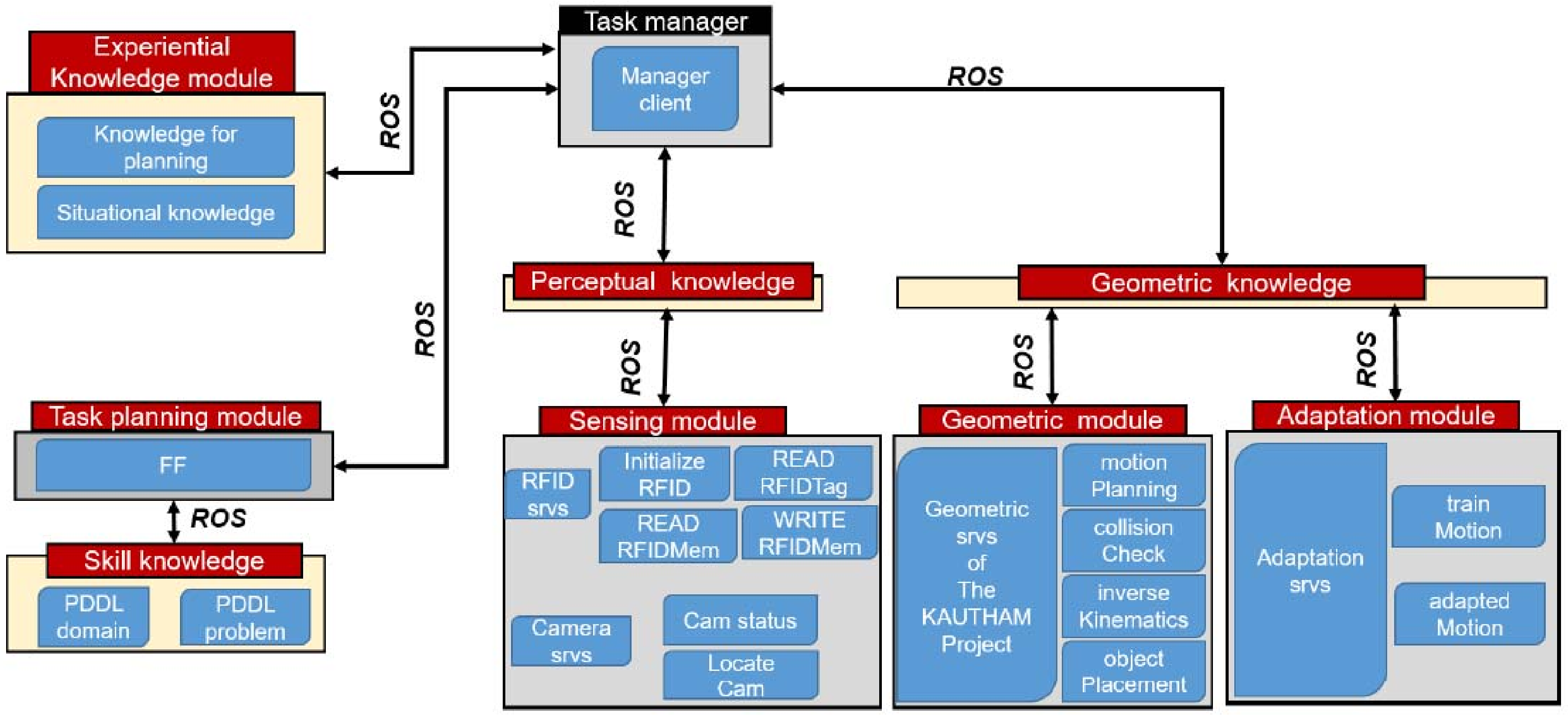
Dynamic Movement Primitives (DMP)

Provides adaption of the robot paths to the actual situations

4. Reasoning for adaptation
4.1. Framework

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning

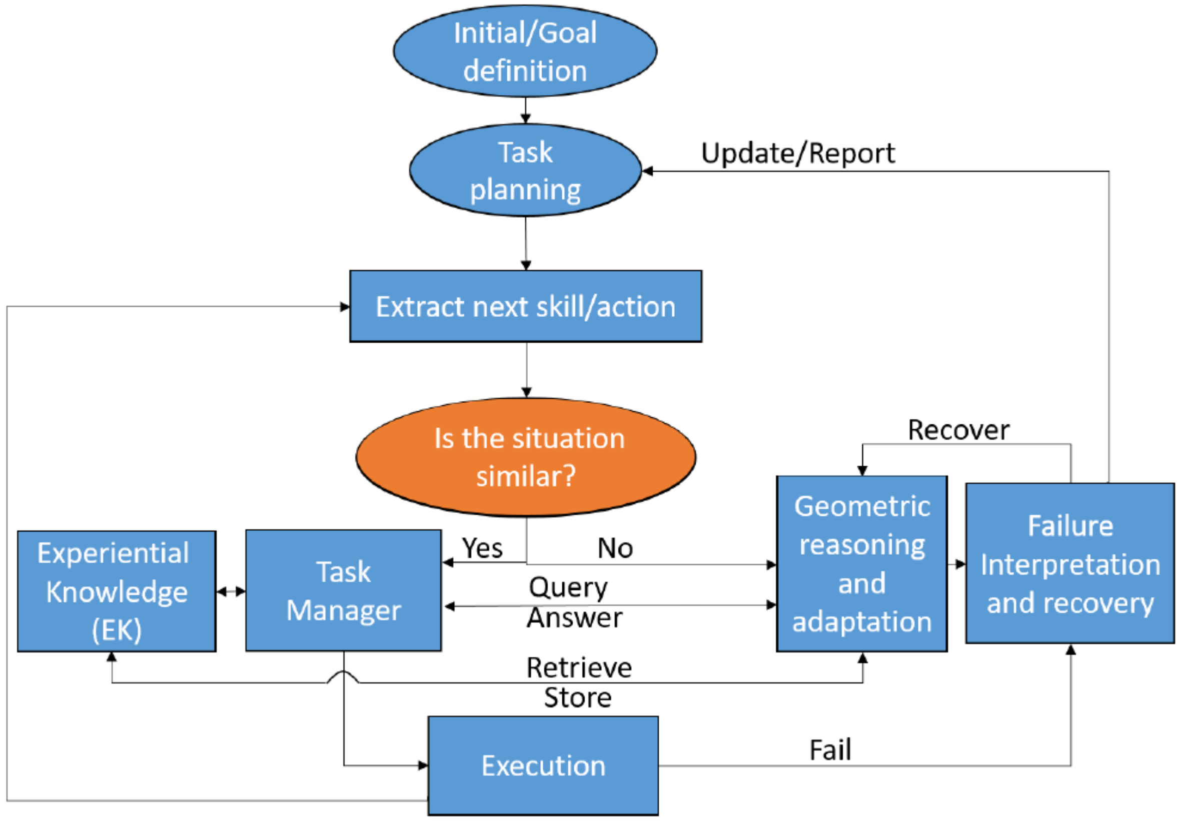
Implementation schema



4. Reasoning for adaptation
4.1. Framework

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning

Flowchart



Initial/Goal definition
Task planning
Update/Report
Extract next skill/action
Is the situation similar?
Recover
Geometric reasoning and adaptation
Failure Interpretation and recovery
Experiential Knowledge (EK)
Task Manager
Yes
No
Query
Answer
Retrieve
Store
Execution
Fail

4. Reasoning for adaptation
4.2. Skill knowledge

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning

A skill, in SkillMaN, is a description of what the robot can do. It is divided into:

1. **Primitive skills** consist of a sequential list of atomic actions, which refers to a single action or gesture, including its preconditions and effects.

2. **Rule-based skills** consist of a set of “if A then B rules” toissue appropriate gestures according to sensors outcome.

Both methods, however, cannot be executed on their own. They require a structure, such as a workflow, that contains the abstract steps that are usually required for task execution.

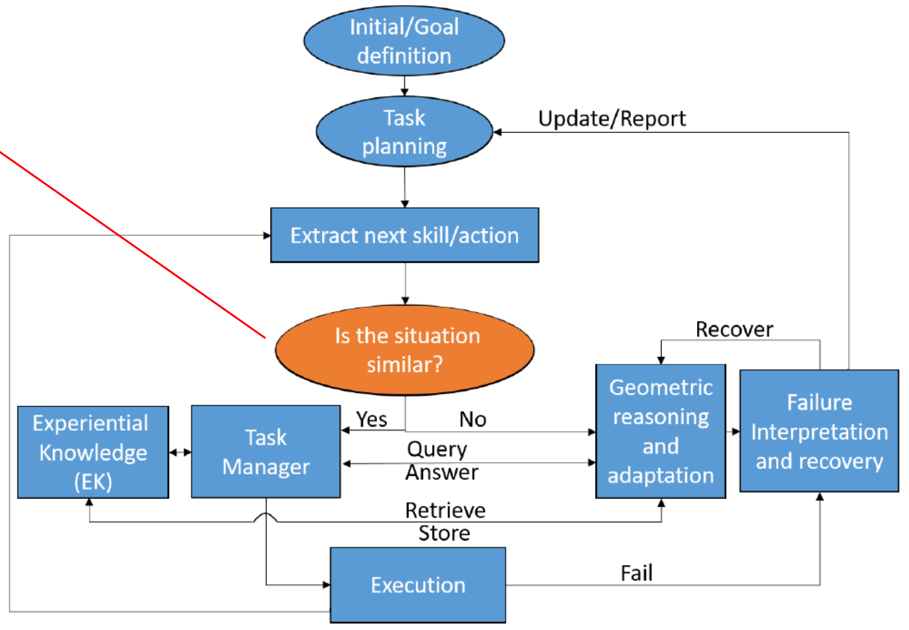
4. Reasoning for adaptation
4.3. Reasoning on situations

SkillMaN: Robotic Manipulation Framework based on Perception and Reasoning

Filtering situation is a process of finding those situations that satisfiesa skill description. It means the robot has to detect the situations that use a specific skill in their description.

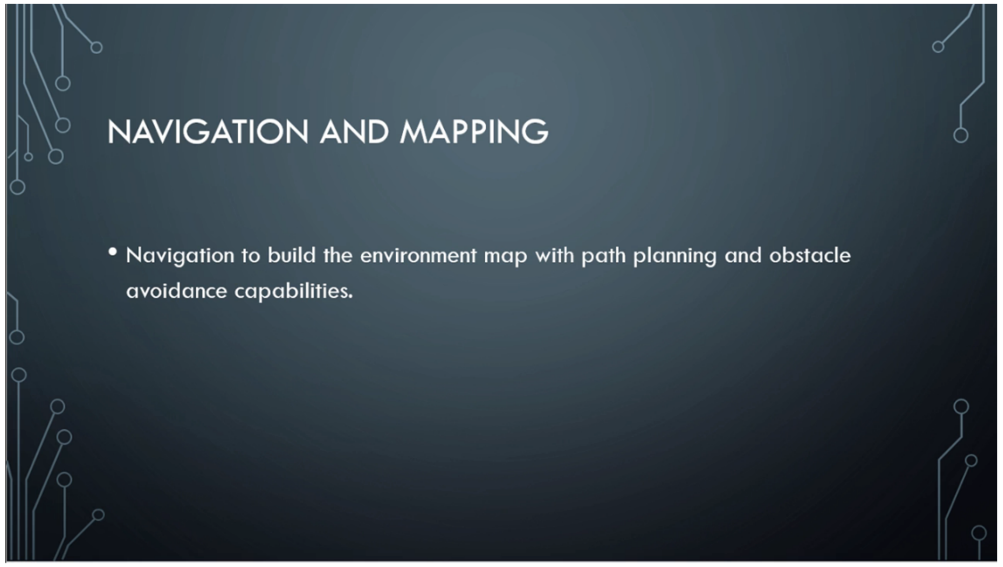
which are the situations that contain a certain skill?

```
?– filterSituation( hasSkill(Situation, Skill) ),
?– filterSituation( hasParticipant(Situation, Object) ),
?– filterSituation( hasPart(Situation, Event) ).
Situation=[Skill, Object, Event].
```



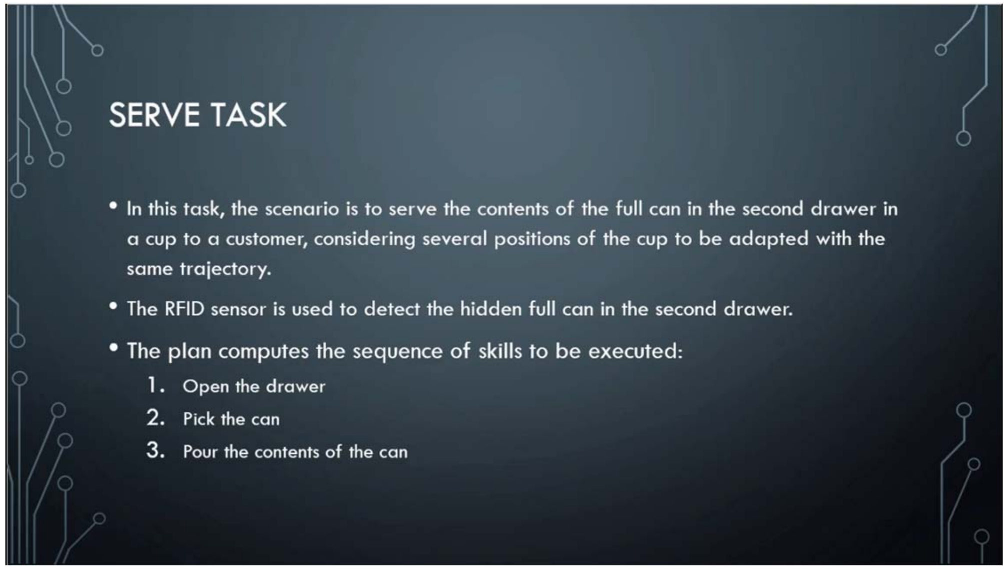
Initial/Goal definition
Task planning
Update/Report
Extract next skill/action
Is the situation similar?
Recover
Experiential Knowledge (EK)
Task Manager
Yes
No
Query
Answer
Geometric reasoning and adaptation
Failure Interpretation and recovery
Retrieve
Store
Execution
Fail

4. Reasoning for adaptation
4.3. Reasoning on situations



NAVIGATION AND MAPPING

- Navigation to build the environment map with path planning and obstacle avoidance capabilities.



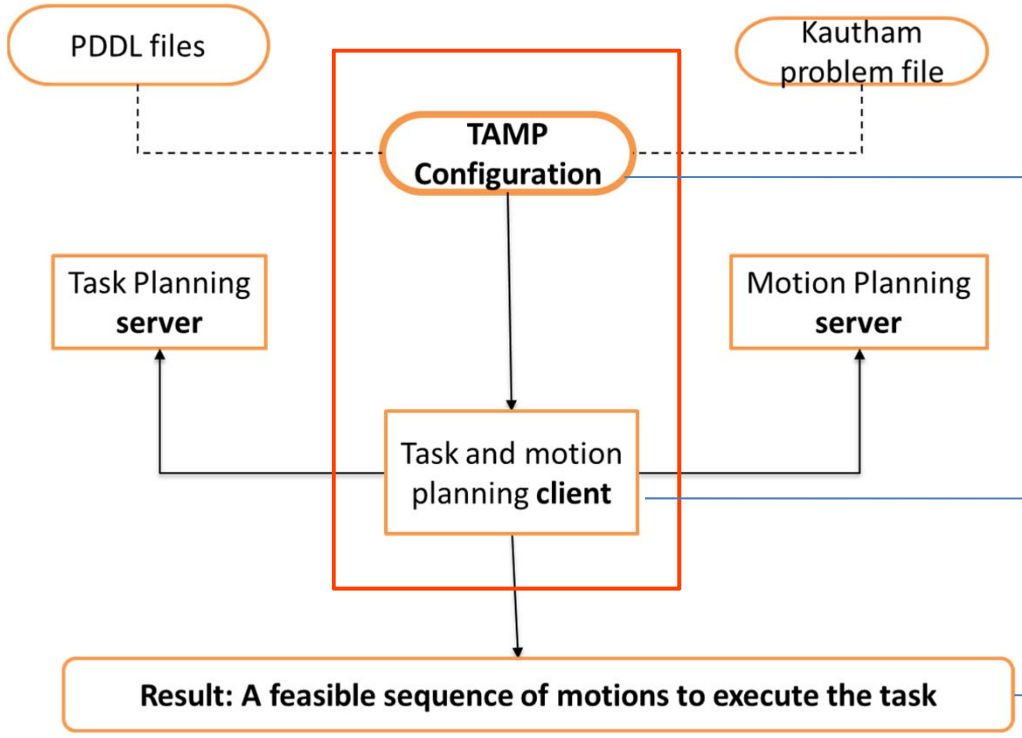
SERVE TASK

- In this task, the scenario is to serve the contents of the full can in the second drawer in a cup to a customer, considering several positions of the cup to be adapted with the same trajectory.
- The RFID sensor is used to detect the hidden full can in the second drawer.
- The plan computes the sequence of skills to be executed:
  1. Open the drawer
  2. Pick the can
  3. Pour the contents of the can

Outline

5. Reasoning for robustness
5.1. Framework

- Automatically generate the code to be executed in the robotic system.
- Be able to adapt to changes in the pose of the objects (geometric-level adaptation)
- Be able to adapt to changes in the location of the objects (task-level adaptation)



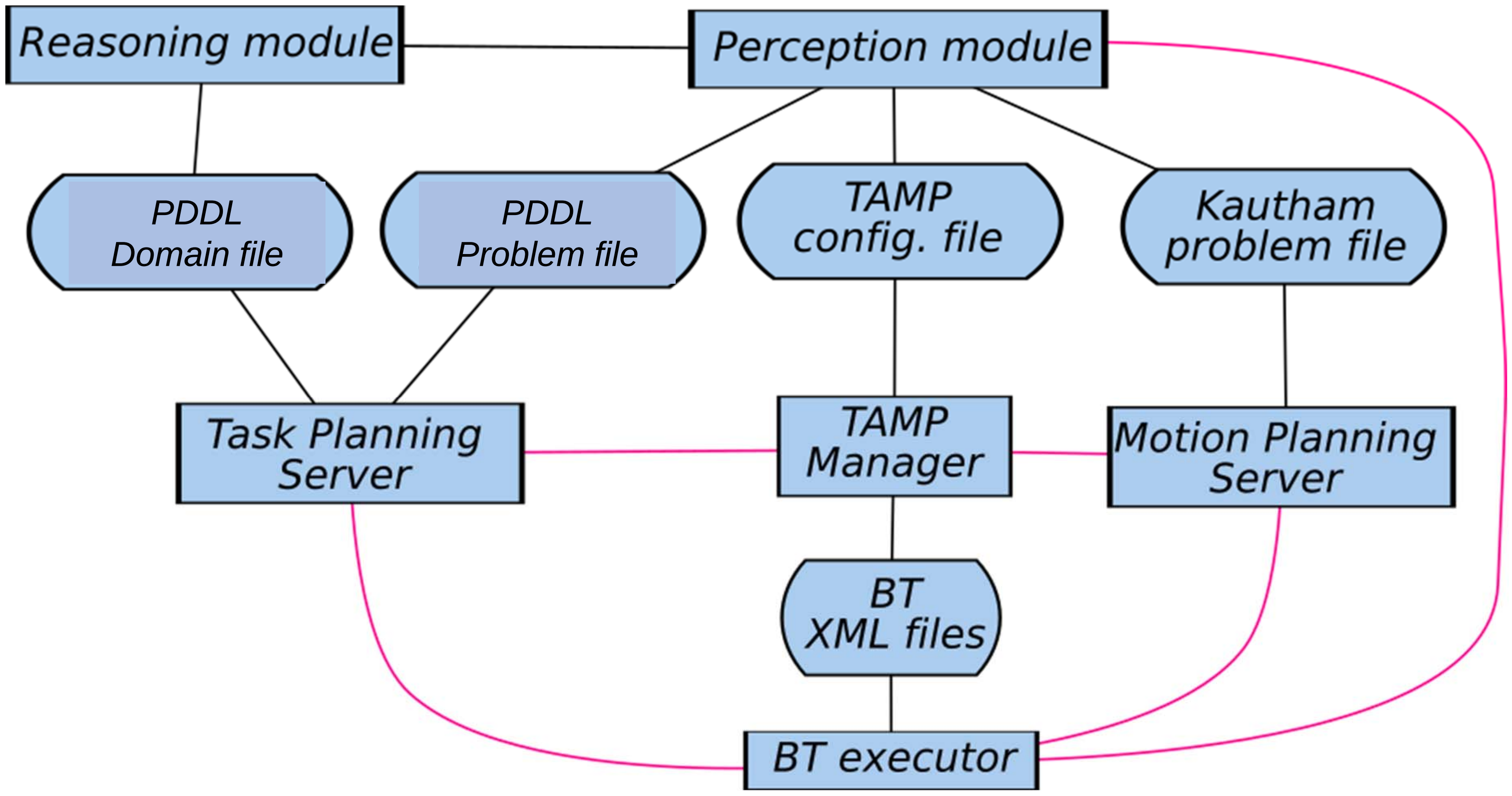
Linking of geometric and symbolic data

Combining planning levels

Automatic writing of XML files representing behavior trees

5. Reasoning for robustness
5.1. Framework

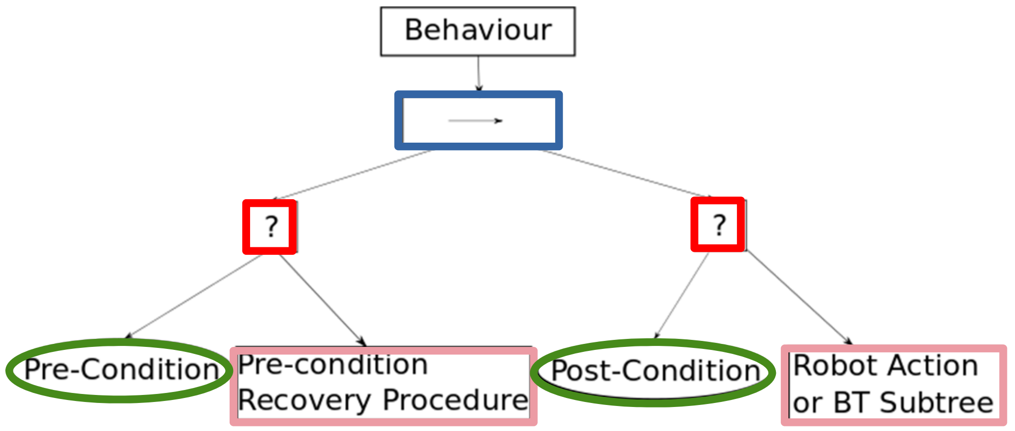
- Automatically generate the code to be executed in the robotic system.
- Be able to adapt to changes in the pose of the objects (geometric-level adaptation)
- Be able to adapt to changes in the location of the objects (task-level adaptation)



5. Reasoning for robustness
5.2. Behavior tree-based execution framework for TAMP

1. The TAMP manger will be responsible of **initially writing the XML** files describing all the BTs needed to perform the manipulation task.
   - A **global task-level BT** for managing the execution at task level
   - A **particular action-level BT per action** for managing the actions of the plan
2. Recovery strategies may be called that **rewrite the BTs**.

All BTs will have the following **general structure**:

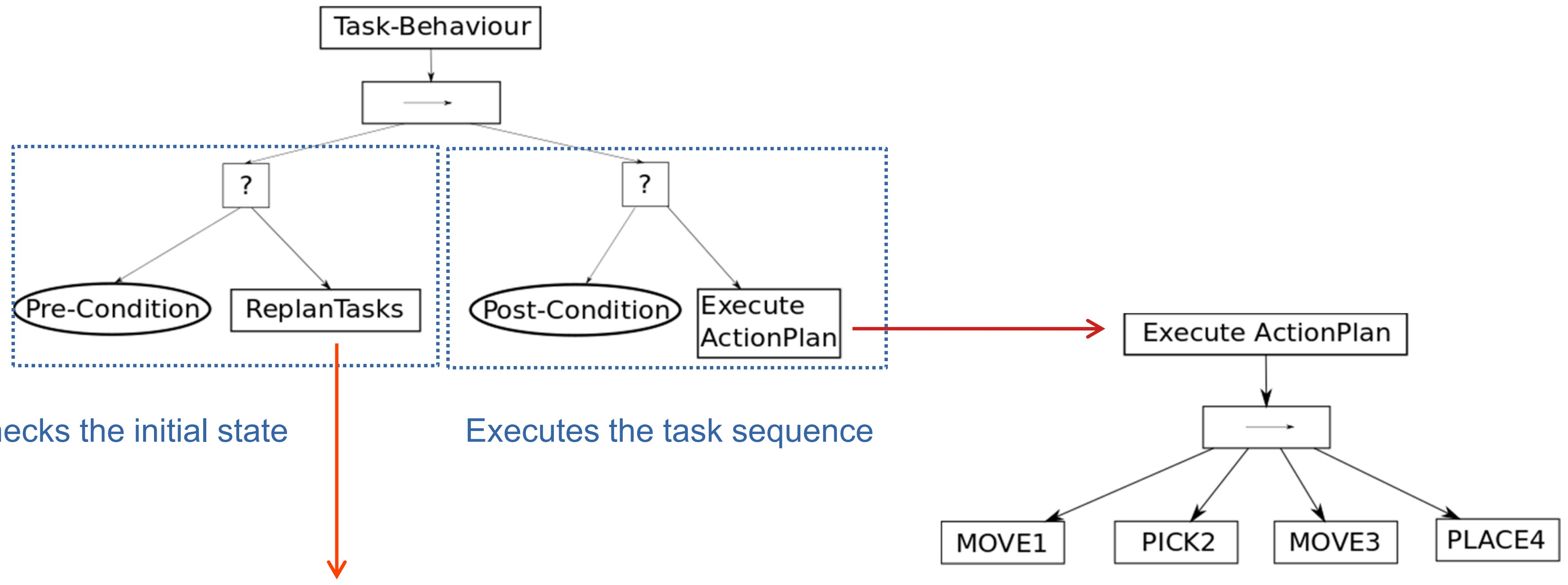


Control Nodes control the execution flow
- Sequence nodes (return success only if all of its child nodes return success)
- Fallback nodes (return success as soon as one of the children returns success)

Execution Nodes: used to query the robot hardware (sensing / actuation)
- Condition nodes (cannot be preempted)
- Action nodes (can be preempted)

5. Reasoning for robustness
5.2. Behavior tree-based execution framework for TAMP

Task-Level behavior: The main BT



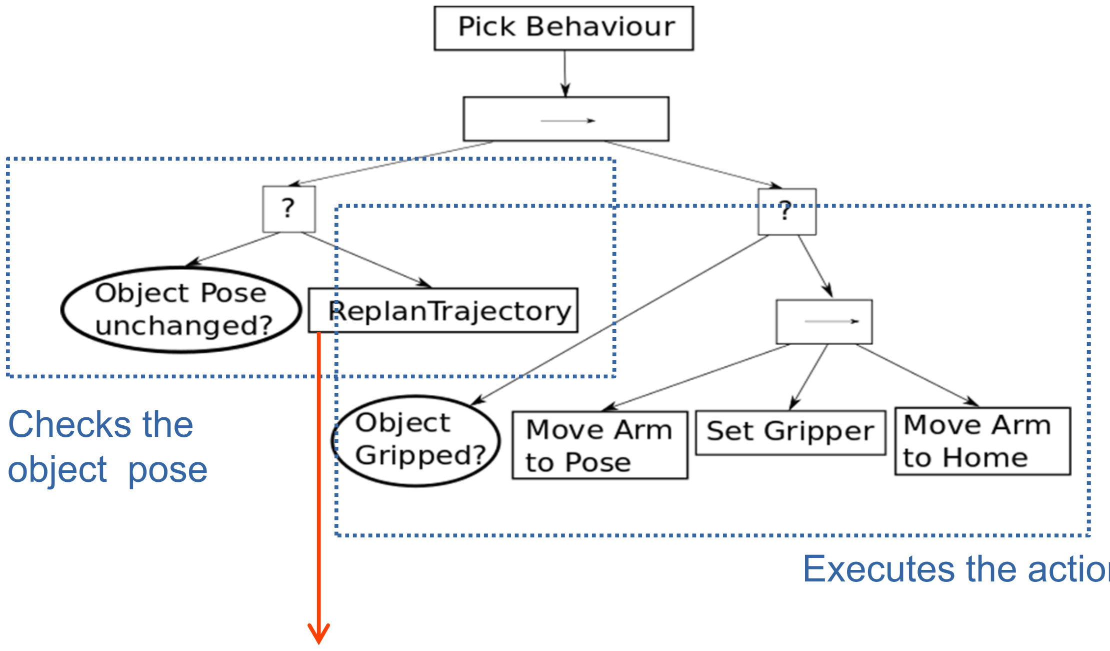
Checks the initial state

Executes the task sequence

Call again the **task** planner from the current state.
TASK-LEVEL ADAPTATION

5. Reasoning for robustness
5.2. Behavior tree-based execution framework for TAMP

Action-Level behaviors: BTs for task actions



Checks the object pose

Executes the action

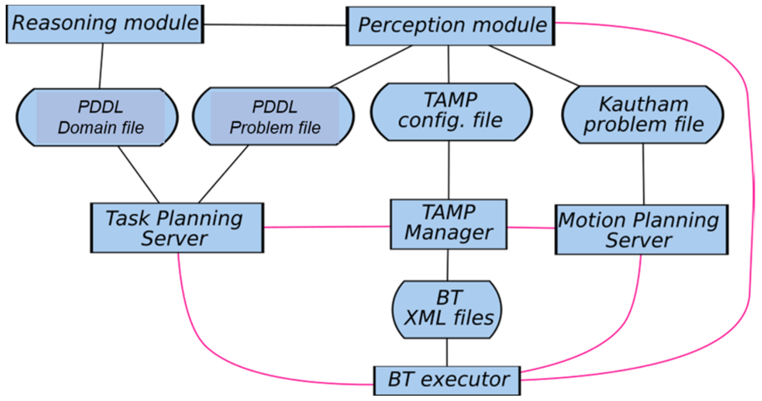
Call again the **motion** planner from the current state.
MOTION-LEVEL ADAPTATION

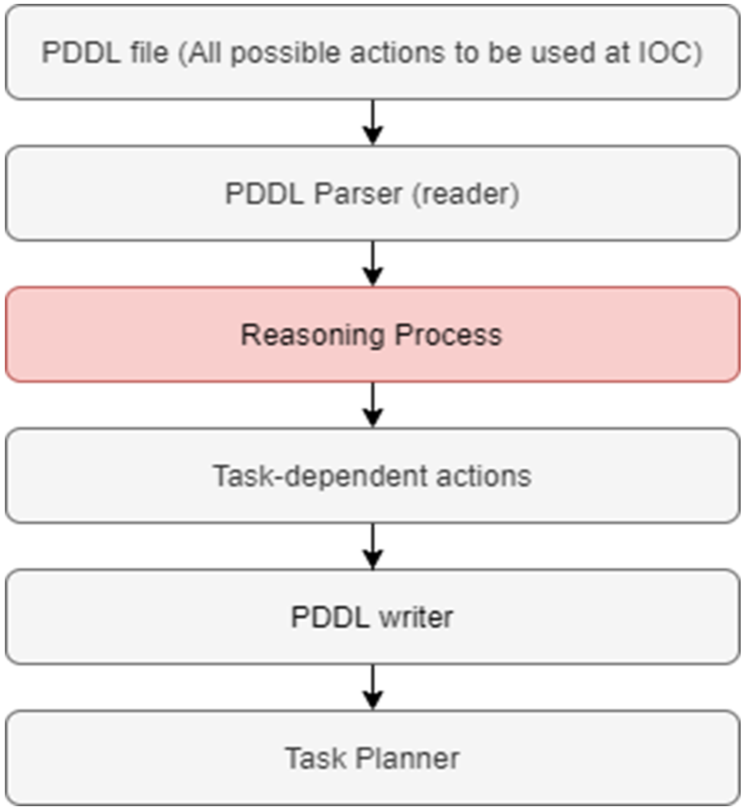
Automatic generation and execution of XML files

1. The Task-Level Behavior BT is generated with the general structure.
2. The *task planner* service is called.
3. The Execute ActionPlan BT with the particular task plan is written
4. For each action:
    1. The *motion planner* service is called and the resulting path is coded as a ROS trajectory as required by the ROS action service.
    2. The corresponding Action-Level Behavior BT is written.
    3. The action nodes are filled with the robot trajectories.
5. The BTs are passed to the BT executor.
6. BTs are rewritten if a state is detected which does not correspond to the expected one.

5. Reasoning for robustness
5.3. Reasoning on actions

The **reasoning module** of the framework will select the required actions to solve the task according to the task goal, the robot capabilities and the initial state.



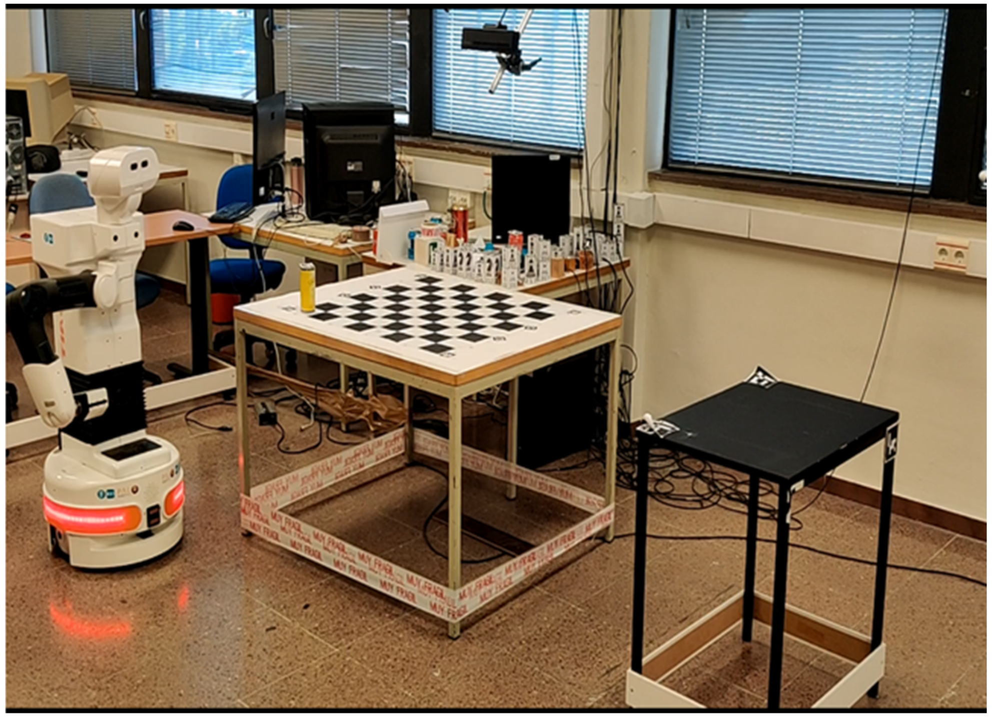


5. Reasoning for robustness
5.3. Reasoning on actions

Questions like:

1. Is there a human present in the environment?
2. Does the robot have navigation capabilities?

have been asked and used to filter the actions in the global PDDL file.



Outline

6. Conclusions and future work

Conclusions

- This talk has presented different approaches to use reasoning to enhance TAMP methods for robotic manipulation.

  Reasoning has improved:

  1. Perception capabilities
  2. Planning capabilities
  3. Adaptation capabilities
  4. Execution capabilities

- Reasoning has helped TAMP to make robots smarter and aware of the environment, of the task goals and of themselves, more adaptive to changes and more confident in being able to complete the task.

6. Conclusions and future work

Current work:

- Broaden the set of actions used in the BT framework and experiment in real scenarios involving different robots (Yumi, TIAGo).
- Within the BT framework integrate:
  - the failures detection and recovery ontology,
  - the motion adaptation procedures.

Future work:

- Use learning techniques to automatically generate manipulation knowledge.

Reasoning for Robot Manipulation

Jan Rosell and Mohammed Diab

Institute of Industrial and Control Engineering (IOC)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)

Institut d'Organització i Control de Sistemes Industrials