Towards Hierarchical Plan-based Robot Control

ICAPS-2020 online summer school course, Oct. 15, 2020

Joachim Hertzberg, Oscar Lima Carrion, Sebastian Stock

German Research Center for Artificial Intelligence (DFKI)

Plan-Based Robot Control research department
Osnabrück, Germany

Course structure

German Research Center for Artificial Intelligence

1. What is plan-based robot control?
2. ROS and ROSPlan – short reminder
3. HPBRC – Integration into ROSPlan
4. Q+A

Course structure

1. **What is plan-based robot control?**
2. ROS and ROSPlan – short reminder
3. HPBRC – Integration into ROSPlan
4. Q+A

Robot plan – historical (and still useful) definition

“The plan is that part of the robot’s program, whose future execution the robot reasons about explicitly.”

Drew McDermott, 1992

img.: Yale University Web pages (CS Emeritus Faculty)

McDermott, D. (1992) Robot planning. AI Magazine, 13(2):55–79

What are robot plans for?

- Not only navigation, but action planning in general (“mission planning”)
- Optimize overall performance
- Make robot performance coherent over longer periods of time
- Allow for user interaction on “natural” abstraction/granularity level
- Enable learning on high granularity levels
- Help structure the control code (software technology argument)

Examples

- “Why are we going this way?”
- “Could we drop by a bank?”

… and BTW, a **robot** is a machine that acts in closed-loop control in its environment, of which it has imperfect control and/or imperfect knowledge&perception.

Robot plans need to be hybrid!

German Research Center for Artificial Intelligence

- Space, time, and "causation" (action dependencies) interact in PBRC
  - Clutter on the table influences the best serving position,
    which influences best grasp, arm trajectory, and arm to use (left, right),
    which influences the table approach trajectory and its duration
    which influences driving parameters (speed, curvature, …),
    which influences high-level actions to employ
    (e.g., carry objects on tray),
    which influences …
- Separating these different planning realms costs optimality and flexibility
- Integrating them creates complexity;
  luckily, **robot plans are short** (or so they should be)

"Waiter Robot" (PR2) example
(EU project RACE, 2011-14)

Planning in robots – you use (need?) it all!

German Research Center for Artificial Intelligence

From ICAPS 2021 CFP (selection)

- Uncertainty and stochasticity in planning and scheduling
- Partially observable and unobservable domains
- Conformant, contingent and adversarial planning
- Plan and schedule execution, monitoring and repair
- Continuous planning, on-line and real-time domains
- Plan recognition, plan management and goal reasoning
- Continuous state and action spaces based planning
- Action model learning, knowledge acquisition and engineering
- Human computer interaction for planning and scheduling systems
- Mixed initiative planning and scheduling systems
- Planning and decision support for human-machine teams
- Human-aware planning and behavior prediction

… are ideally all needed in PBRC, and all in integration!

Fundamental questions in PBRC

- How weigh, at any time, the influences of reactions/reflexes (may kick in at any time) and of the planned action just under execution?
- How feed necessary information in symbol form (“state description”) to the planner, in real time? (e.g., object anchoring)
- How deal (in the planner? in the KB system?) with contradictions resulting from sensor/interpretation errors and from different data recency?
- How recognize/determine at execution time when an action is finished (success – failure – abandoned)? Challenge: **non-nominal execution**!
- What are plan formats supporting (both?) plan generation and monitored execution?
- How manage goals to achieve by the robot, aided by the PBRC?

… so why hierarchical plans in PBRC?

German Research Center for Artificial Intelligence

- Allow for fade-out effect (cf. “planning in the now”, L. Kaelbling):
  plan in detail for near future, abstract from detail in distant future
  - planning efficiency:
    spend no resources planning in detail for a distant future that may never happen,
    yet let future action influence imminent action

- Recover from execution failure by failing upward:
  change concretion/decomposition of abstract/high-level task under execution,
  while leaving (most of) the rest of the plan intact
  - planning efficiency and plan robustness:
    don’t re-plan from scratch whenever something went differently,
    don’t throw away future plan parts unaffected by short-term action variation

Kaelbling, L.P.; Lozano-Perez, T. 2011. Hierarchical planning in the now. Proc. ICRA-2011

Course structure

1. What is plan-based robot control?
2. **ROS and ROSPlan – short reminder**
3. HPBRC – Integration into ROSPlan
4. Q+A

What is ROS?

- ROS = Robot Operating System
- *Not really an operating system, but is easier to explain in the business world this way...*
- ROS = (A) plumbing + (B) tools + (C) capabilities + (D) ecosystem (Brian Gerkey)
- (A) provides publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed computing systems.

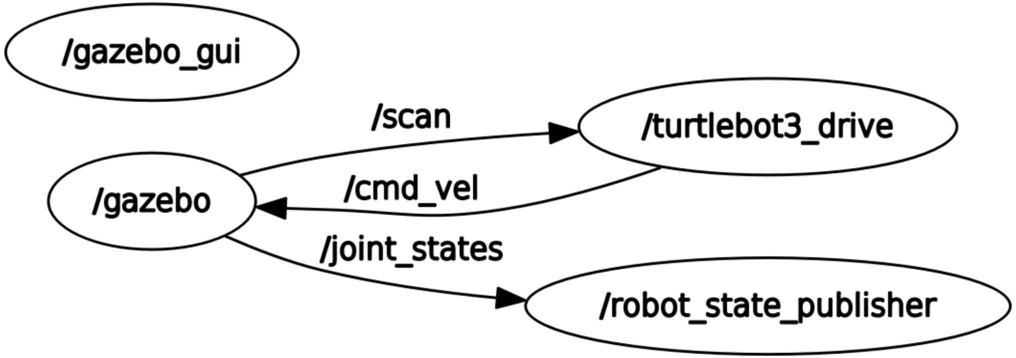
- Source: https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/

What is ROS? (A) plumbing + (B) tools + (C) capability + (D) ecosystem

- (B) ROS provides an extensive set of tools for configuring, starting, introspecting, debugging, visualizing, logging, testing, and stopping distributed computing systems.
- (C) ROS provides a broad collection of libraries that implement useful robot functionality, with a focus on mobility, manipulation, and perception.
- (D) ROS is supported and improved by a large community, with a strong focus on integration and documentation. ros.org is a one-stop-shop for finding and learning about the thousands of ROS packages that are available from developers around the world.

Basic concept 1 : Node

- **Node : a process that performs computation**
- Nodes are combined together into a graph and communicate with one another using topics, services, actionlib and the Parameter Server
- Nodes may reside in different machines transparently



Basic concept 2: Topic

- Topics: Named buses (pipes) over which nodes exchange messages.
- Producer – Consumer design pattern
- In general, nodes are not aware of who they are communicating with.
- nodes that are interested in data ***subscribe*** to the relevant topic.
- nodes that generate data ***publish*** to the relevant topic.

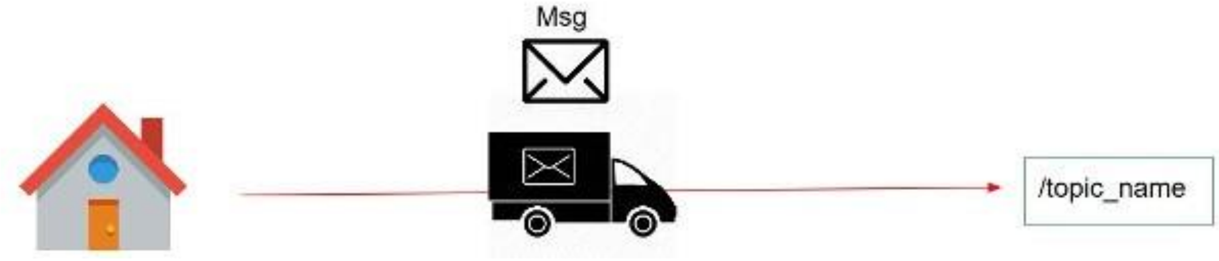
Properties of topics

German Research Center for Artificial Intelligence

- There can be multiple publishers and subscribers to a topic.
- Intended for unidirectional, streaming communication.
- Nodes communicate with each other by publishing **messages** to topics.
- An active topic can only have a single **message** type at a time.

Basic concept 3: Messages

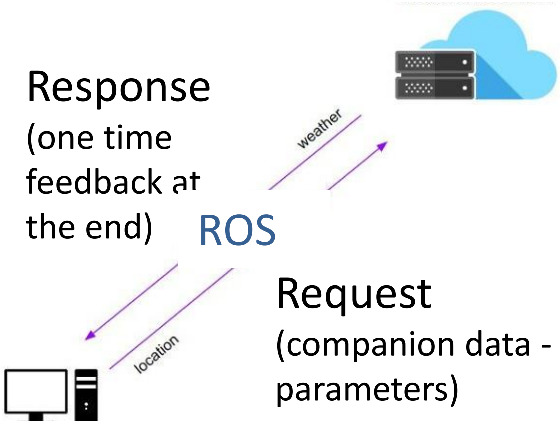
- Earlier we saw that nodes can make connections with each other via topics.
- Informally the message can be seen as the envelope you send via post.
- Messages state what kind of information your nodes need to produce in order to communicate together.



Basic concept 4: Service

- Service: is a client/server communication request system.
- There is no feedback while the operation is being performed but only one time at the end.

server



Performs operation

Response (one time feedback at the end)

ROS

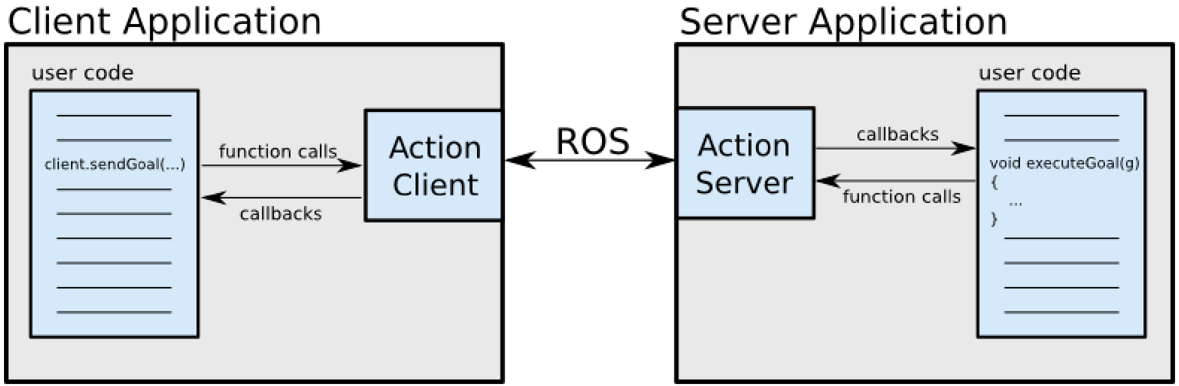
Request (companion data - parameters)

client

Typically is a blocking operation, but you can specify a timeout...

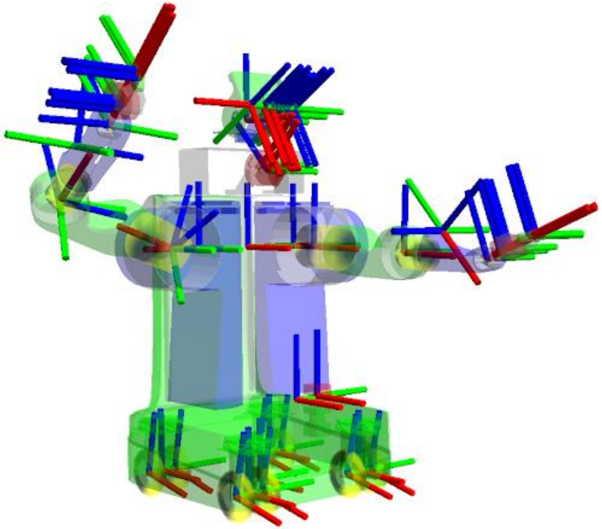
Basic concept 5: actionlib

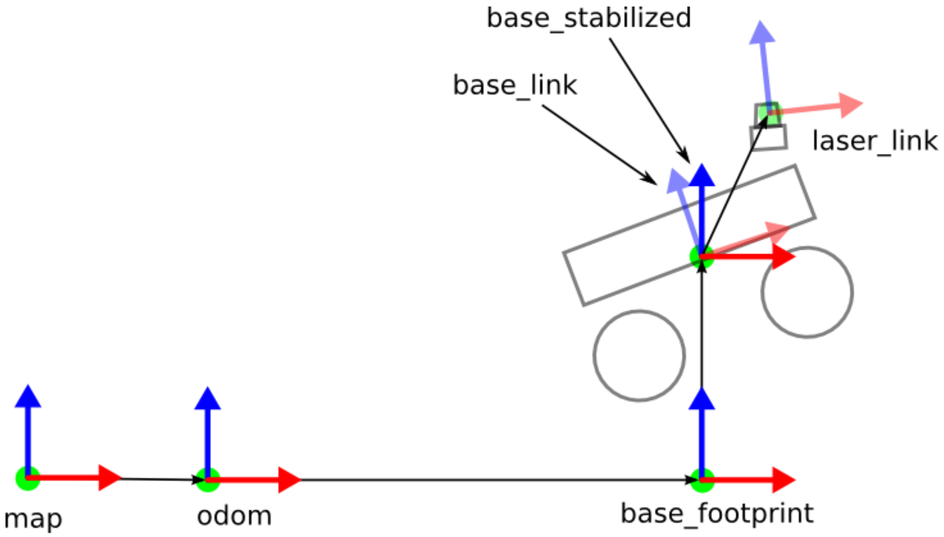
- Similar to services but additionally you can:
  - Get periodic feedback about the progress of the request
  - Temporarily interrupt a task being carried out
  - Cancel the request



Some important libraries 1/4

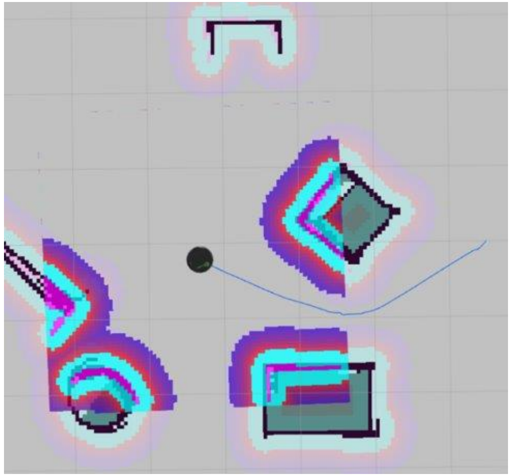
- **tf** : Library to keep track of multiple coordinate frames over time.
- Users can transform points, vectors, etc between any two coordinate frames at any desired point in time.

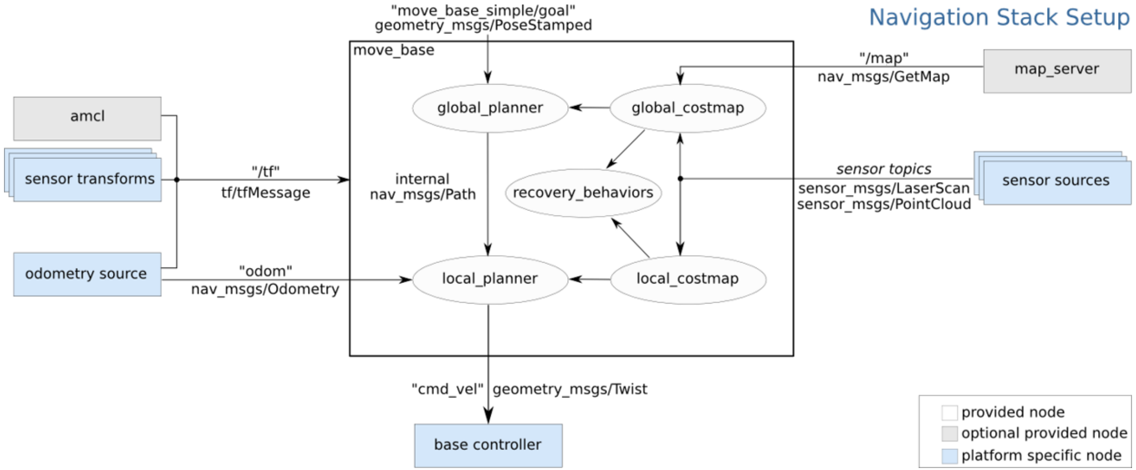




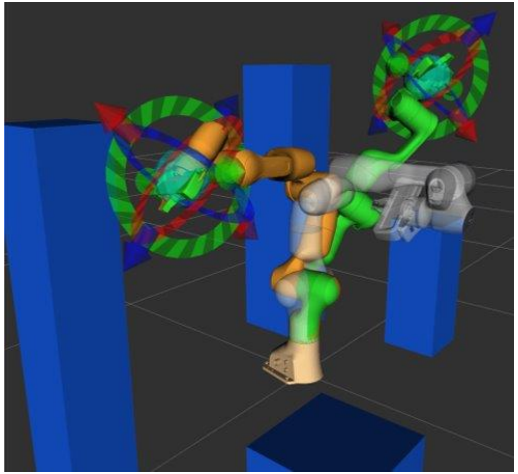
Some important libraries 2/4

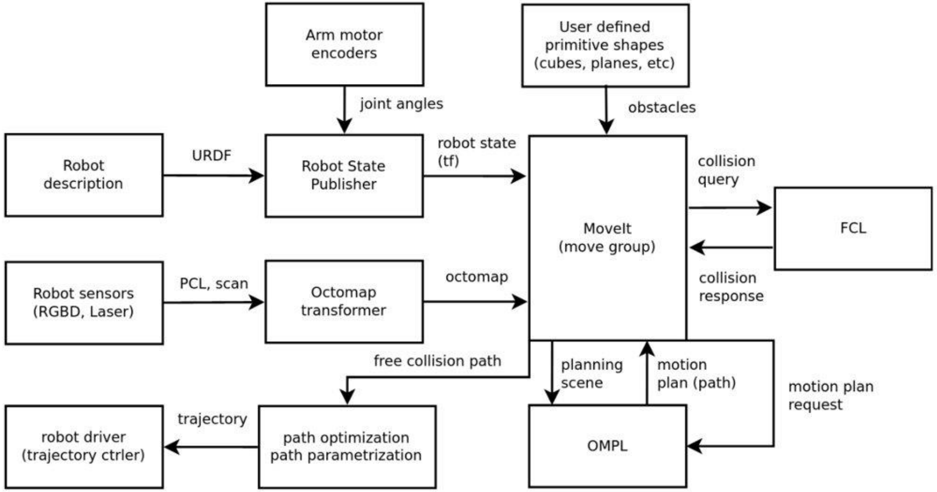
- **move_base**





Some important libraries 3/4

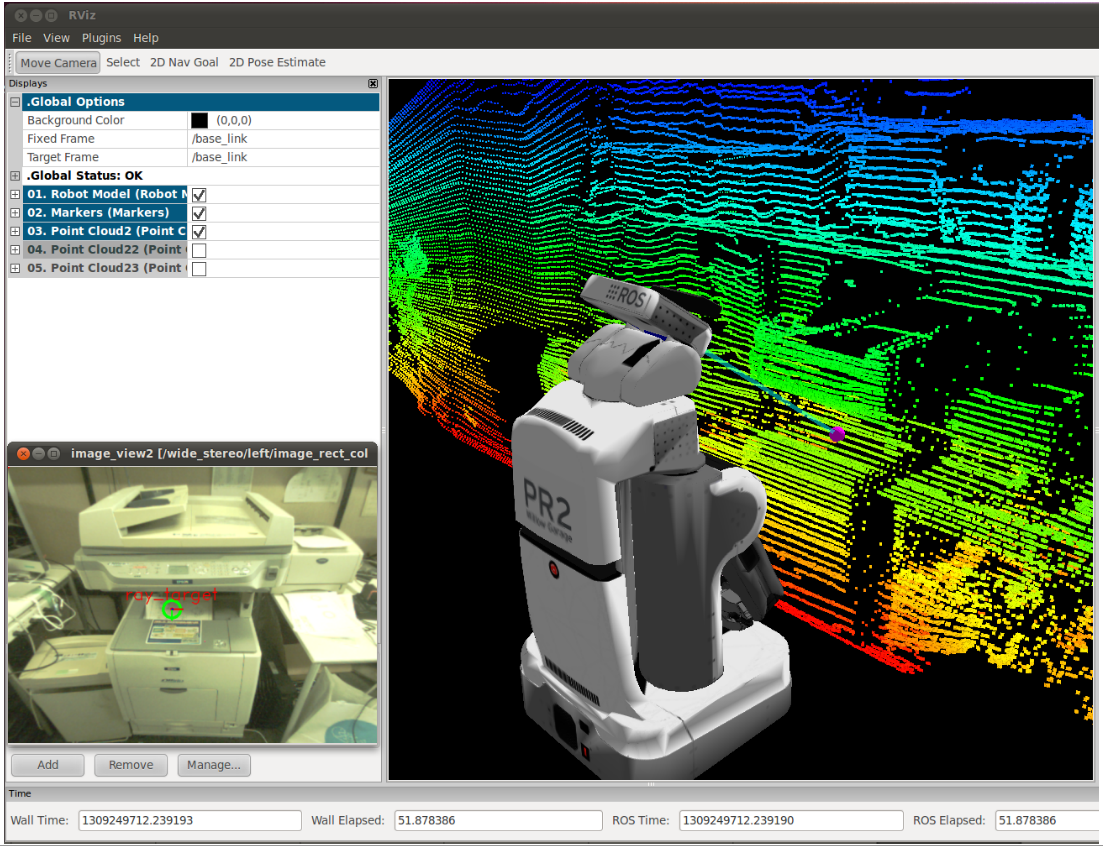




Some important libraries 4/4

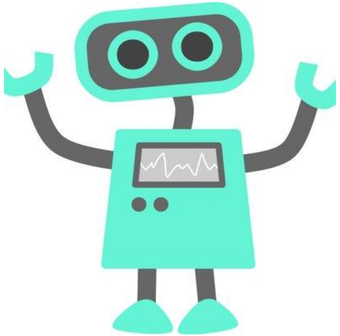
- **Rviz**

3D visualisation tool for ROS

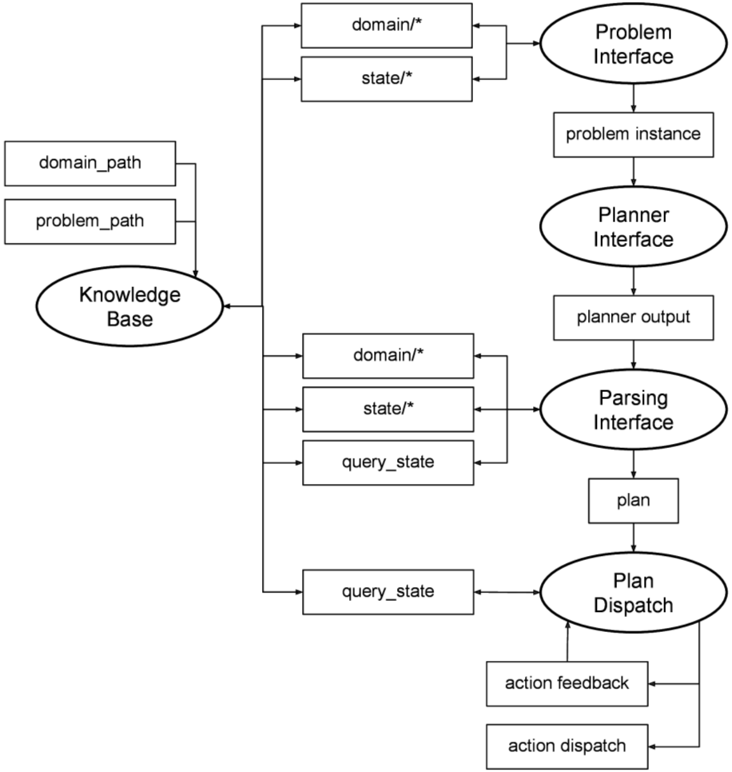


ROSPlan

- *"The ROSPlan framework provides a generic method for task planning in a ROS system".*
- First step towards integration of AI planning and robotics.
- Uses different technologies to provide with high level robot control.
- Main support: PDDL 2.1
- Experimental support: PPDDL, RDDL, HDDL (**HTN** new!)
- Plan execution and monitoring via:
  - Simple plan dispatch or Esterel plan dispatch



Architecture

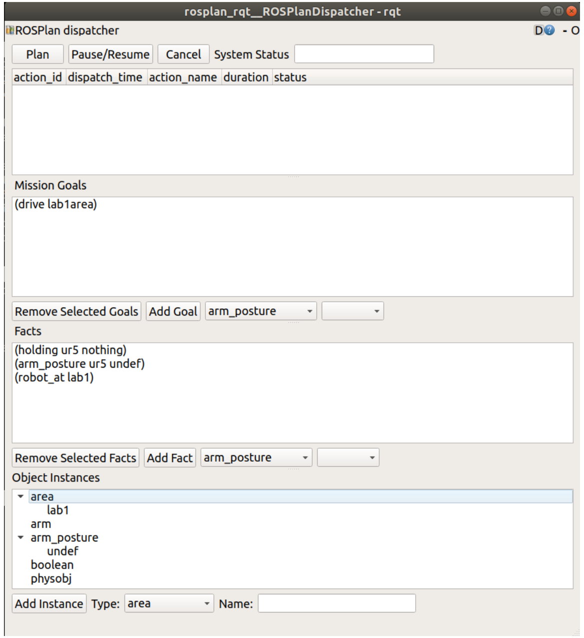


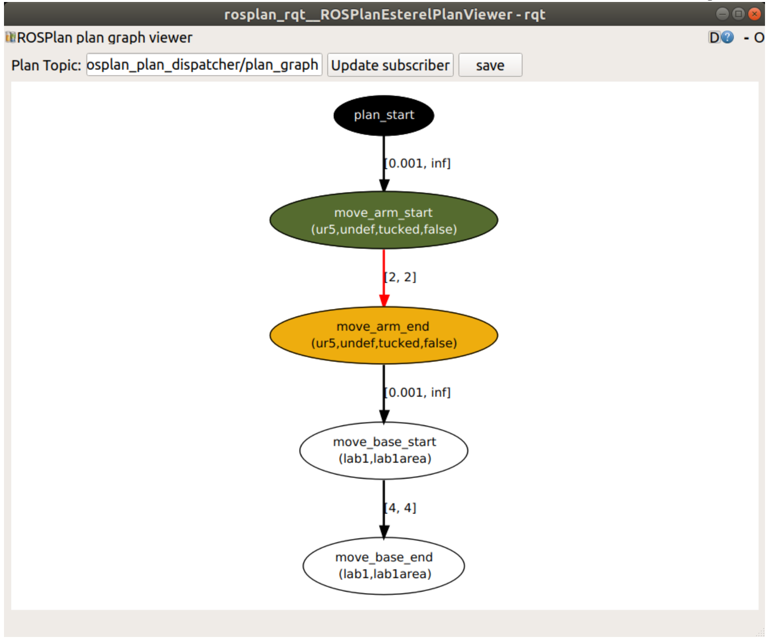
Components

- Knowledge Base (KB) : stores the (symbolic) planning model (domain and state); Communication via services
- Problem interface : query state from KB and create a problem instance
- Planner interface : wrapper around the AI planner, write its output to a topic
- Parsing interface : Convert planner output into a representation suitable for execution
- Plan dispatch : Execution and monitoring layer

Debugging tools

- KB rqt gui : KB visualisation/manipulation
- Esterel rqt gui : Plan visualisation and realtime execution progress





Action interface

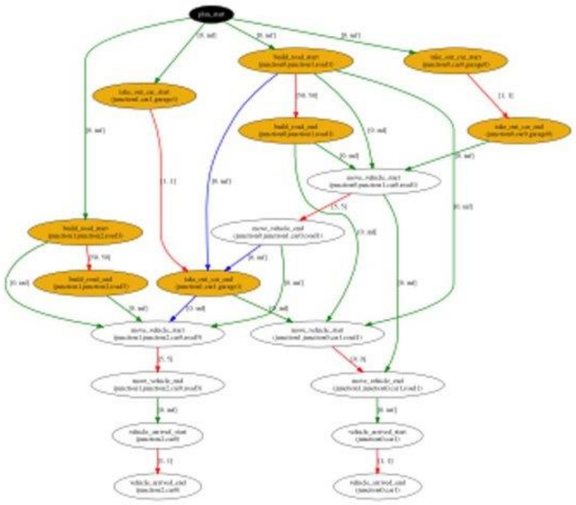
German Research Center for Artificial Intelligence

- Provides a base class implementation to ease the process of robot action creation
- Available in C++ and Python
- Steps:
- 1) Subscribe to the plan
- 2) If action is not relevant, exit
- 3) Send actionlib feedback telling the action is enabled
- 4) Provide virtual function for concrete implementation
- 5) Upon action success, update KB according to the model
- 4) Send actionlib result (action achieved or failed)

Esterel plan dispatch

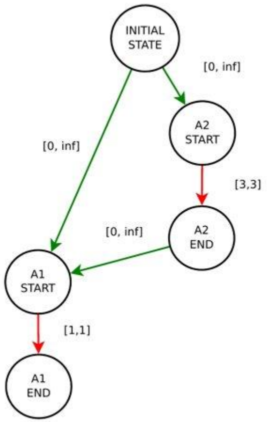
- Realtime graph-based algorithm for plan execution - monitoring
- Support for concurrent actions
- Preconditions are checked before sending action for execution

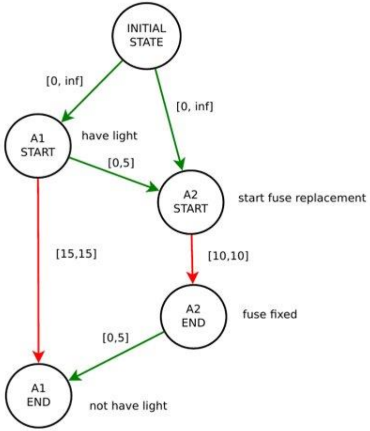
- In a nutshell: Converts a temporal PDDL planner output into a graph, which edges represent ordering constraints.



Esterel plan dispatch, Semantics behind edges

- Conditional edge encapsulates 1 or more casual links
- All edges specify ordering constraints: source node effects need to
- happen before sink node gets signal
- Node cannot fire unless it has received all incoming edges / signals
- Examples:





Simulated Actions

German Research Center for Artificial Intelligence

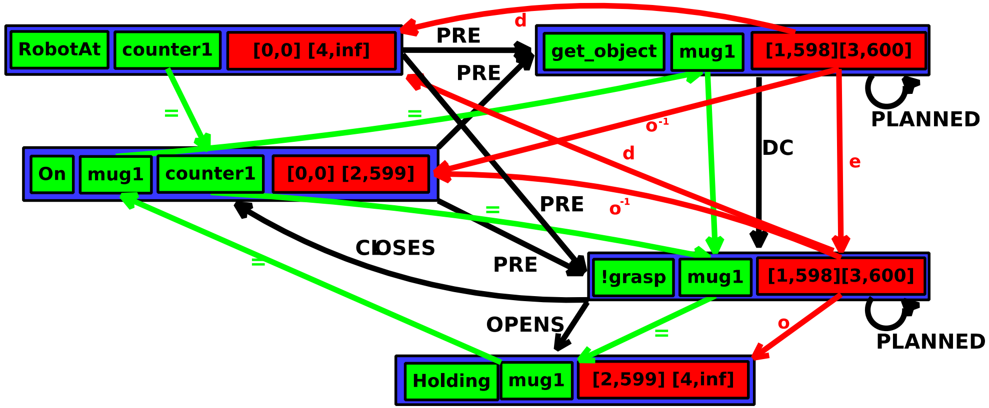
- Syntethic simulator of actions
- Made as a replacement of a physics-based robot simulator
- Make multiple mockup actions, useful for testing/debugging
- Parameters:
  - action_duration
  - action_duration_stddev
  - action_probability

Course structure

1. What is plan-based robot control?
2. ROS and ROSPlan – short reminder
3. **HPBRC – Integration into ROSPlan**
4. Q+A

CHIMP

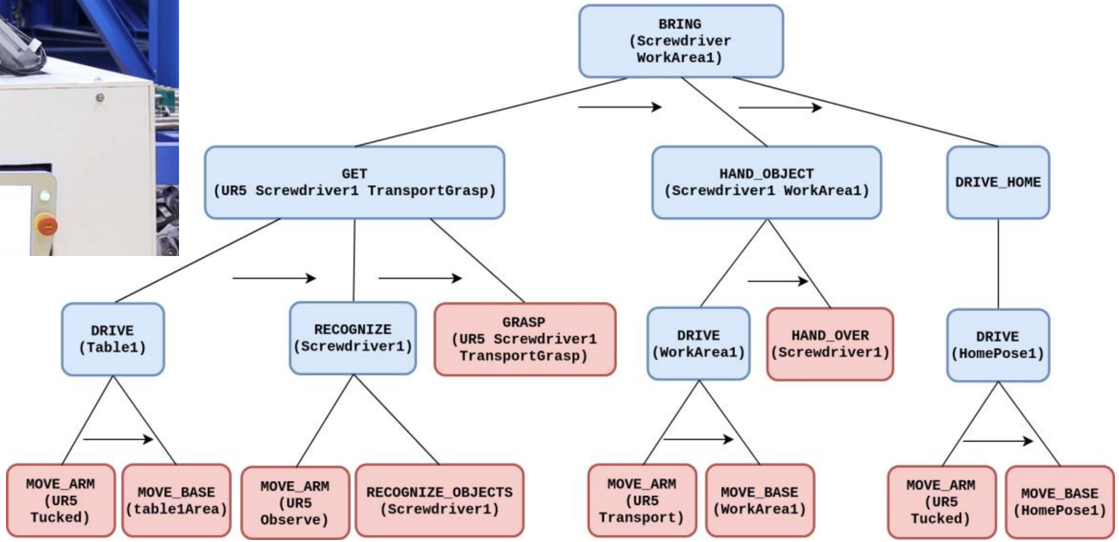
- Hierarchical hybrid task planner that combines
  - Causal/task knowledge
  - Temporal knowledge
  - Resource requirements
  - Integer constraints
- Plan-execution
- Re-planning



(EU project RACE, 2011-14)

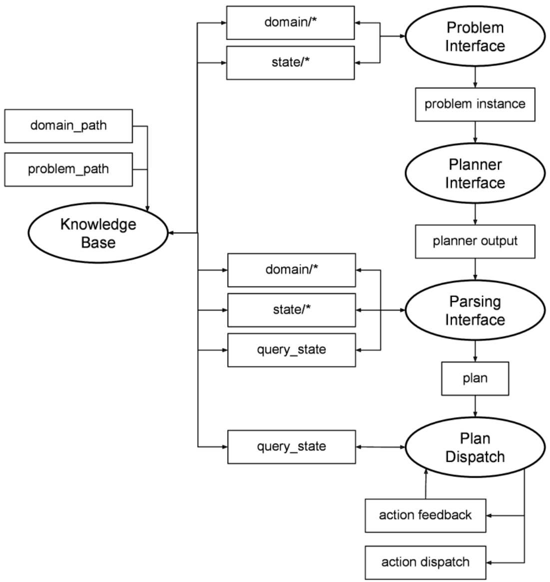
https://github.com/sebastianstock/chimp

CHIMP – Example



Towards Hierarchical Planning in ROSPlan

- ROSPlan was built for PDDL-planners
  - e.g., Knowledge-Base (including message interface) is structured for PDDL-domains
- Goals:
  - Integrate hierarchical planners into ROSPlan
    … without having to rebuild all of it
  - Make ROSPlan more generic/independent from ROS:
    Open Task Planning (and Execution) Library
    …



HDDL - Hierarchical Domain Definition Language

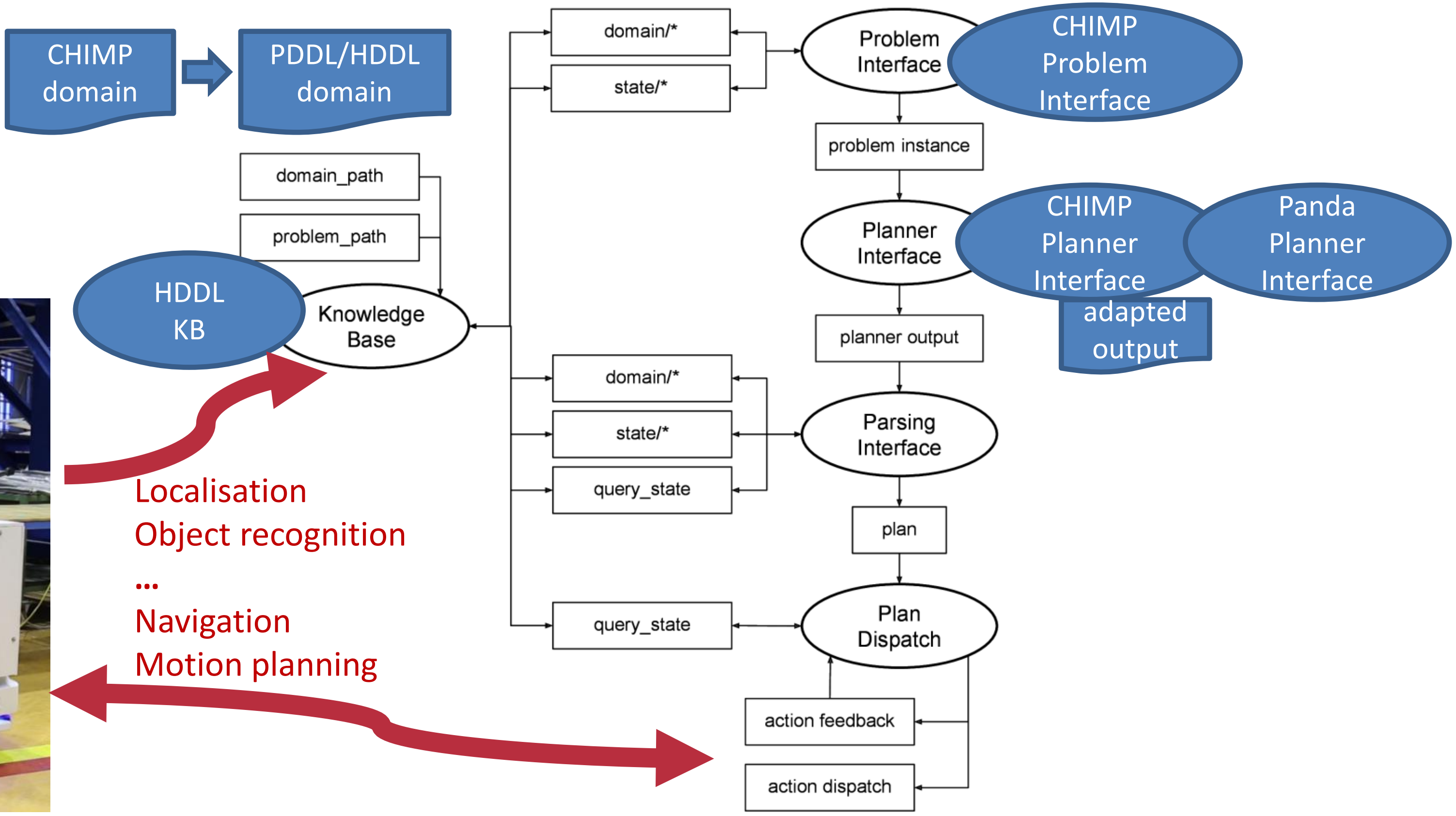
German Research Center for Artificial Intelligence

- Hierarchical planning formalism
- *Höller et al.: HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems (2020)*
- Standard input language for the 2020 IPC for Hierarchical Planning (International Planning Competition)
  - http://gki.informatik.uni-freiburg.de/competition
- Syntax in line with PDDL
- Limitations: no time support, no resources
- Possible alternative: ANML

HDDL (syntax) domain example

```
(define (domain safe_navigation)
  (:requirements :typing)
  (:types
    arm - robot_part
    arm_posture - movement
    area - location
    physobj - object
    boolean - symbol
  )

  (:predicates
    (arm_posture ?arm - arm ?posture - arm_posture)
    (robot_at ?area - area)
    (holding ?arm - arm ?obj - physobj)
  )

  (:task drive :parameters (?from_area ?to_area - area))

(:method tuck_and_move
  :parameters (?from_area ?to_area - area ?arm - arm ?old_posture
               ?new_posture - arm_posture ?keep_gripper_orientation - boolean)
  :task (drive ?from_area ?to_area)
  :ordered-subtasks (and
    (move_arm ?arm ?old_posture ?new_posture
              ?keep_gripper_orientation)
    (move_base ?from_area ?to_area))
)

(:action move_arm
  :parameters (?arm - arm ?old_posture ?new_posture - arm_posture
               ?keep_gripper_orientation - boolean)
  :precondition (and
    (arm_posture ?arm ?old_posture))
  :effect (and
    (not (arm_posture ?arm ?old_posture))
    (arm_posture ?arm ?new_posture))
)
```

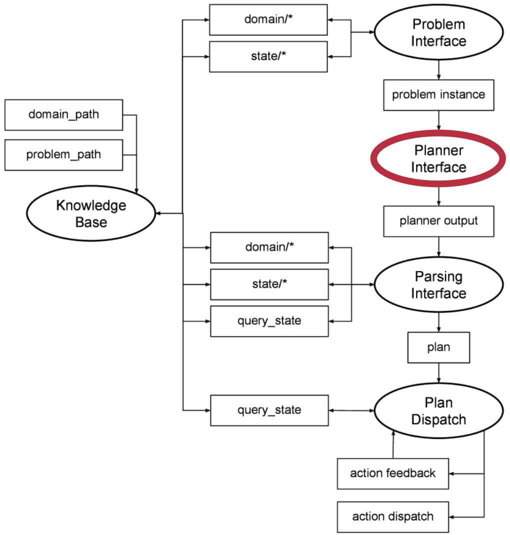
Integrating Hierarchical Planners



Planner Interfaces for CHIMP and HDDL

Wrapper around the planners:

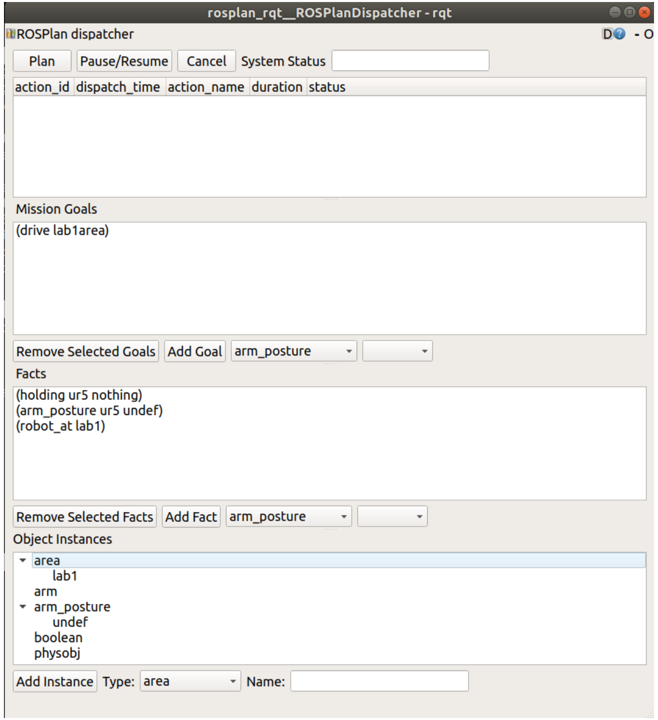
- Write problem to file
- Call planner via command line interface
- Publish relevant part of the plan on the `planner_output` topic
- CHIMP's output format has been adjusted to match the output format of POPF, allowing it to be processed by the parsing interface



HDDL Knowledge Base

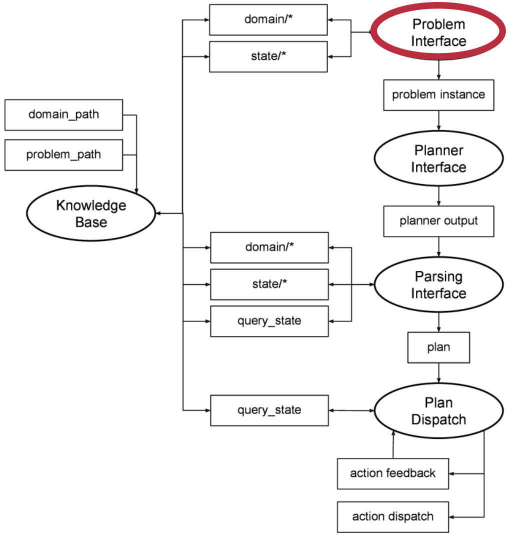
- Uses HDDL parser to load the domain
- Stores the HDDL domain model and current state
- Access via service calls
- Limitations:
  - Can't store methods yet
  - Can't parse HDDL problems
  - Uses experimental HDDL Parser

    https://github.com/oscar-lima/hddl_parser



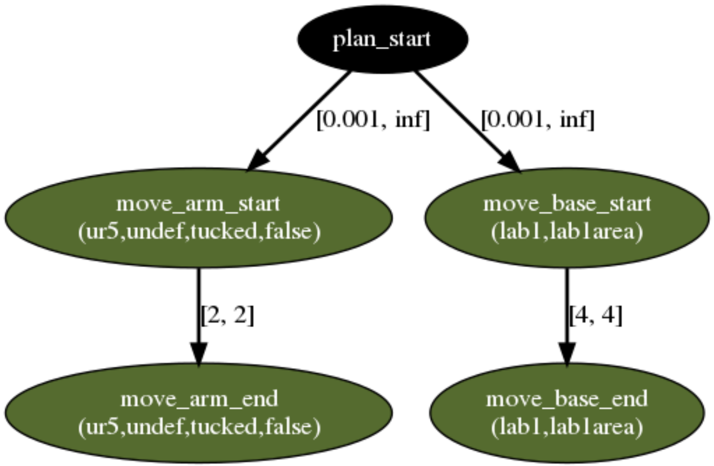
Problem Interfaces

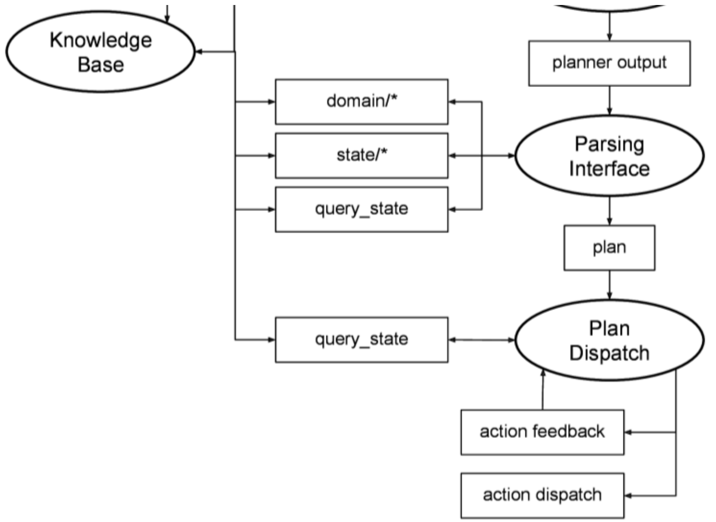
- Uses domain model to query relevant predicates from KB
- Creates problem instance in HDDL or CHIMP format



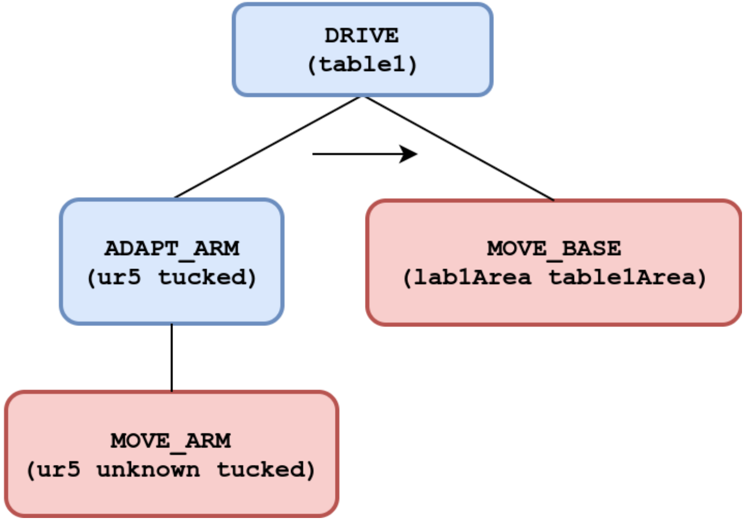
Plan Execution – Esterel Plan Dispatch

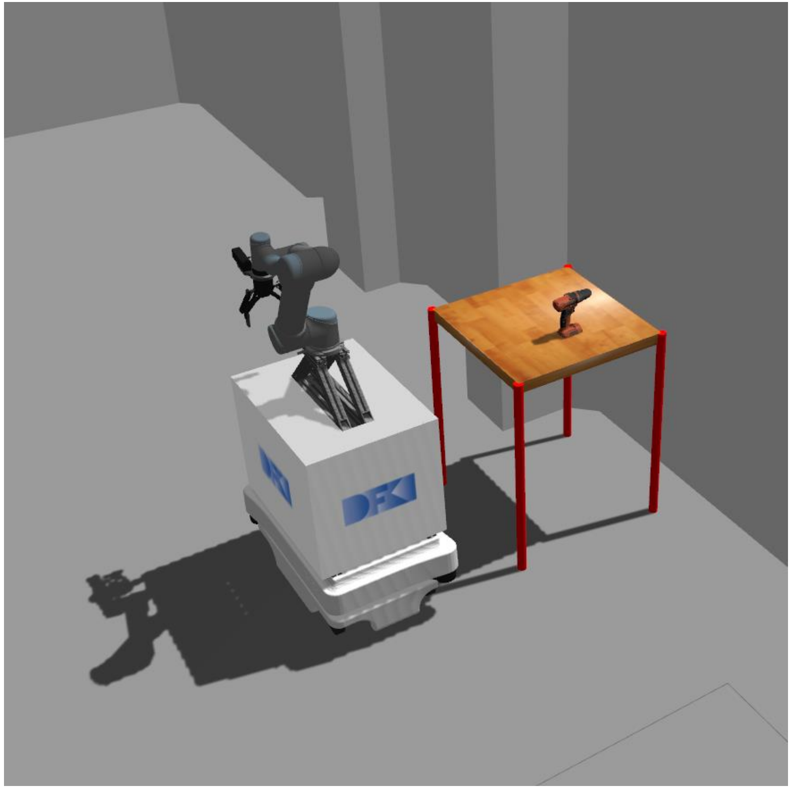
Plan parser creates a graph for dispatching actions:



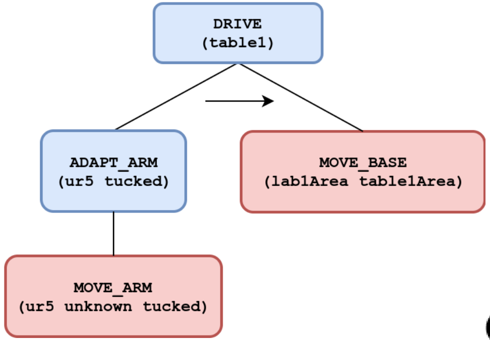


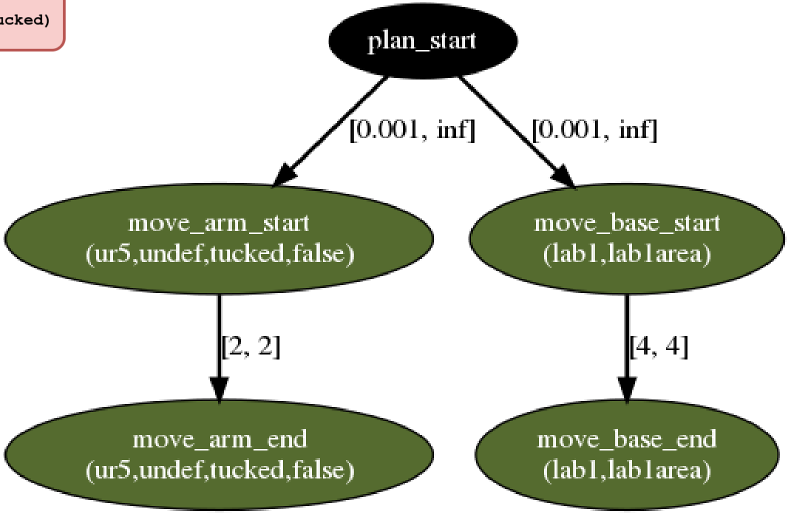
Simple example





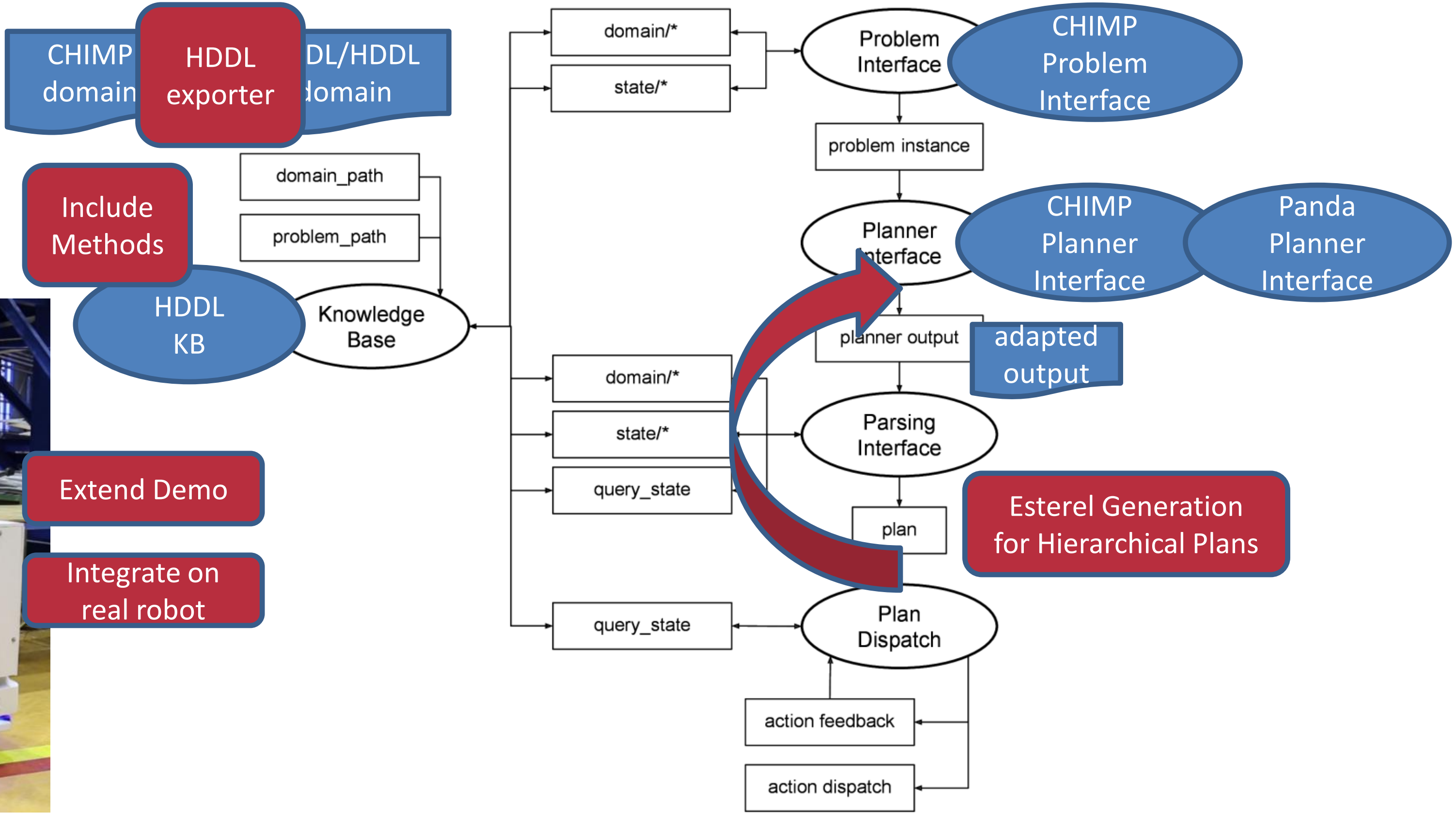
Plan Execution



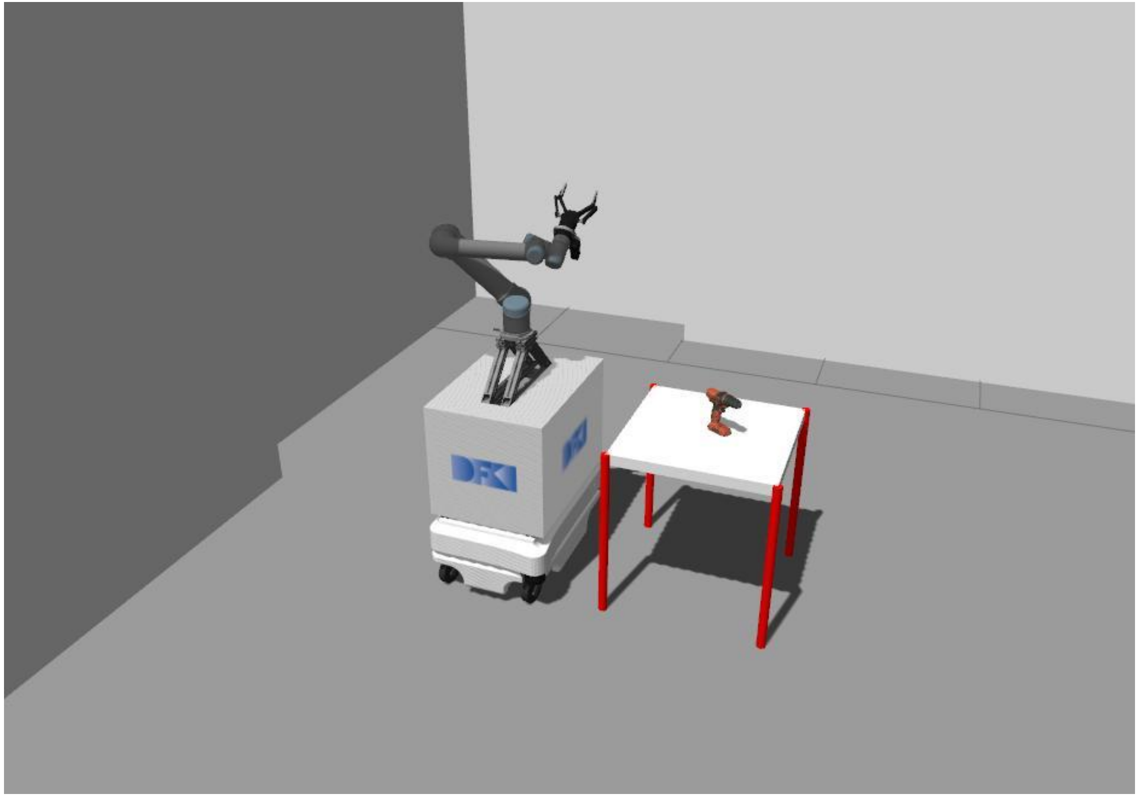



```
(:operator
 (Head move_arm(?arm ?old_posture ?new_posture)
 (Pre p1 arm_posture(?arm ?old_posture))
 (Del p1)
 (Add e1 arm_posture(?arm ?new_posture))

 (ResourceUsage (Usage arm_man_capacity 1))

 (Constraint Duration[2000,INF](task))
)

(:operator
 (Head move_base(?from_area ?to_area))
 (Pre p1 robot_at(?from_area))
 (Del p1)
 (Add e1 robot_at(?to_area))
 (Constraint Duration[4000,INF](task))
)
```

Next steps


CHIMP domain
HDDL exporter
DL/HDDL domain
Include Methods
domain_path
problem_path
HDDL KB
Knowledge Base
domain/*
state/*
Problem Interface
CHIMP Problem Interface
problem instance
Planner Interface
CHIMP Planner Interface
Panda Planner Interface
planner output
adapted output
domain/*
state/*
query_state
Parsing Interface
plan
Esterel Generation for Hierarchical Plans
query_state
Plan Dispatch
action feedback
action dispatch
Extend Demo
Integrate on real robot

Demo



Course structure

German
Research Center
for Artificial
Intelligence

1. What is plan-based robot control?
2. ROS and ROSPlan – short reminder
3. HPBRC – Integration into ROSPlan
4. **Q+A**