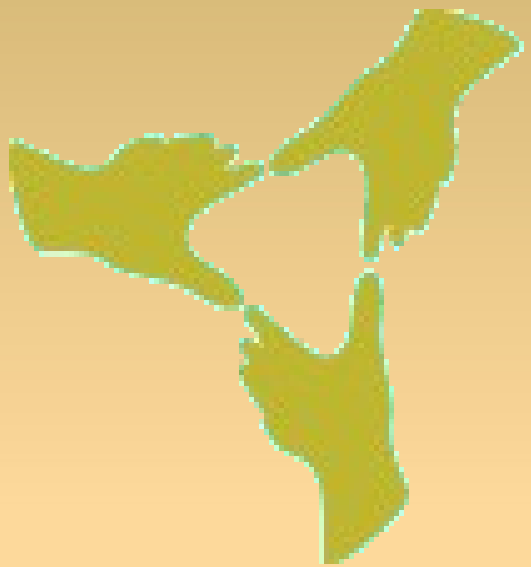
Planning with hybrid knowledge in the GeRT project



GeRT

SEVENTH FRAMEWORK PROGRAMME

www.GeRT-Project.eu

F. Lagriffoul, L. Karlsson, F. Pecora and A. Saffiotti

Center of Applied Autonomous Sensor Systems (AASS)
Örebro university - Sweden

DLR Deutsches Zentrum für Luft- und Raumfahrt e.V. in der Helmholtz-Gemeinschaft
German Aerospace Center (DLR)

ÖREBRO UNIVERSITY
University of Örebro

UNIVERSITY OF BIRMINGHAM
University of Birmingham

Max-Planck-Institut für biologische Kybernetik
Max Planck Institute

The GeRT project

Generalizing Robot manipulation Tasks

Aim: coping with novelty in manipulation tasks.

- Abstract away the variability in objects (shape, size and pose)
- Adapt to the context in which the task is performed (obstacles, initial configuration)
- Perform tasks it has not encountered before, given the necessary building blocks for that task

The planning problem in GeRT

Achieving this involves different research areas:

- Planning

  To compute a sequence of actions to perform the task

- Learning

  To extract knowledge from example programs

- Machine perception

  To figure out the locations and types of objects in the scene

- Grasping

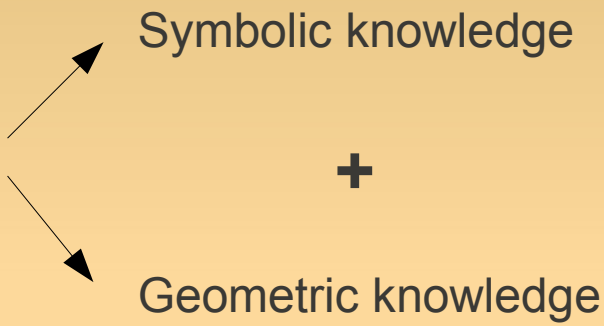
  Which is a difficult problem on its own

The planning problem in GeRT

Achieving this involves different research areas:

- **Planning → Hybrid planning**

  To compute a sequence of actions to perform the task



- Learning

  To extract knowledge from example programs

- Machine perception

  To figure out the locations and types of objects in the scene

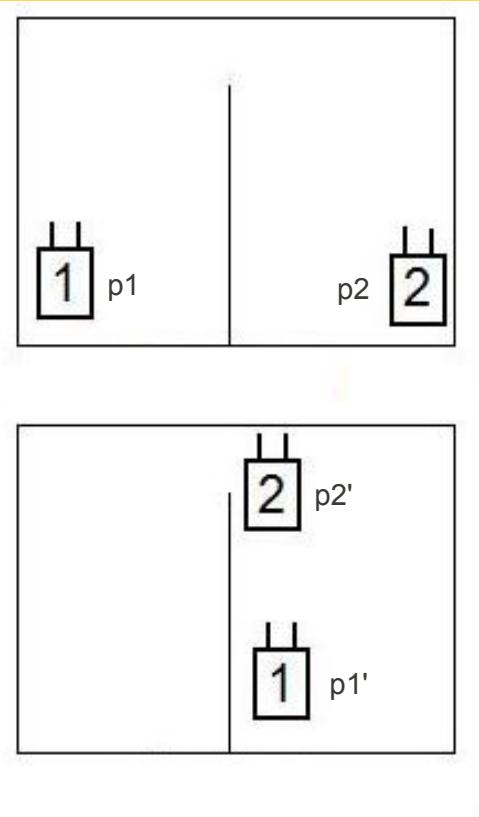
- Grasping

  Which is a difficult problem on its own

The planning problem in GeRT

Hybrid planning

Geometric knowledge **Symbolic knowledge**



<u>Initial state</u>

at robot1 p1
at robot2 p2

<u>Goal state</u>

at robot1 p1'
at robot2 p2'

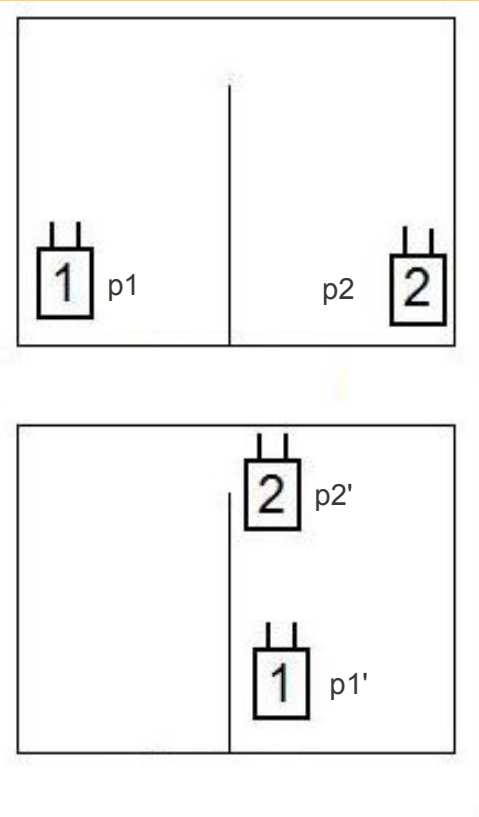
2 plans are solution to the symbolic problem:

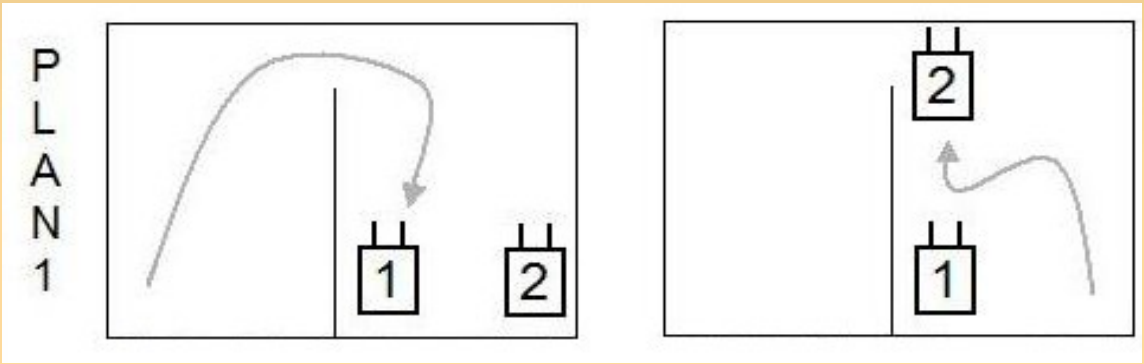
PLAN1:
move robot1 p1'
move robot2 p2'

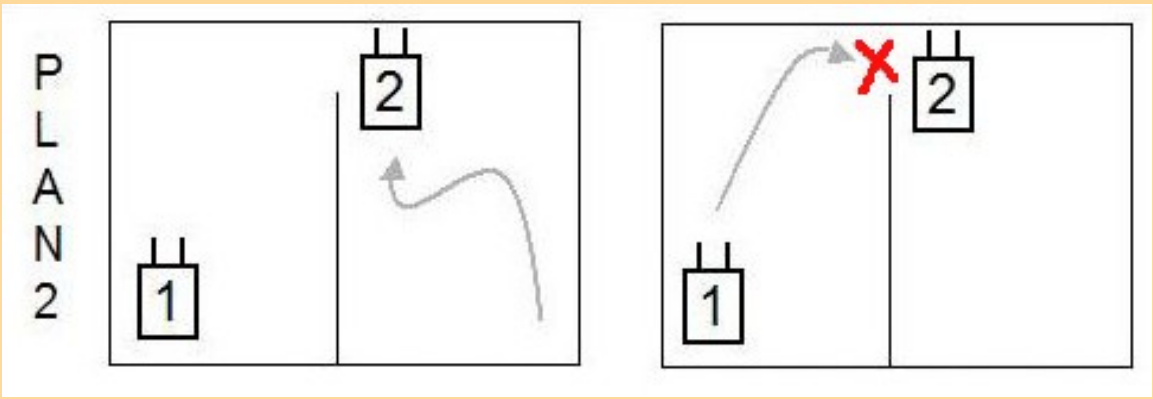
PLAN2:
move robot2 p2'
move robot1 p1'

The planning problem in GeRT

Hybrid planning







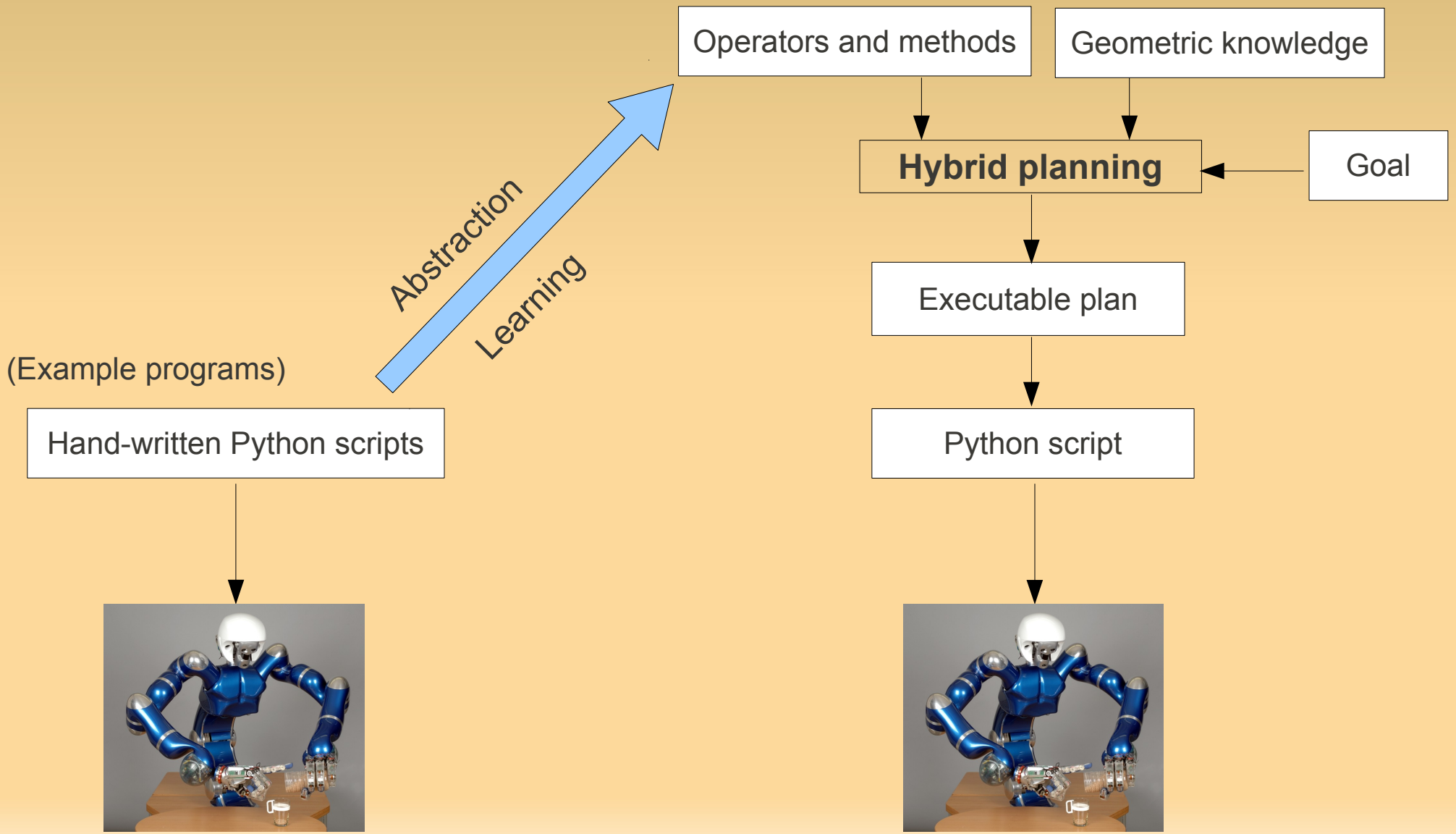
Relation to workshop topics

- **Terminology**: Task descriptions, action recipes, skills and other primitives, and what are their relationships?
- **Conventions**: shared definitions, conventions (coordinate systems, units), and data structures
- **Scene graphs**: How should data be represented (maps, objects, actions)? What data should be annotated and how? What kind of reasoning is performed or needed?
- **Reuse of knowledge**: How can a robot decide which knowledge (e.g., map or skill) to reuse in a new situation?
- **Reuse of tools**: What existing software modules, algorithms, libraries, or APIs can be reused?
- **Knowledge engineering**: How will/should the knowledge base grow? What are the processes leading to creation of a substantial knowledge base useful in real applications, i.e. the bootstrapping of the KB?

Terminology, Action recipes

Today:

GeRT:

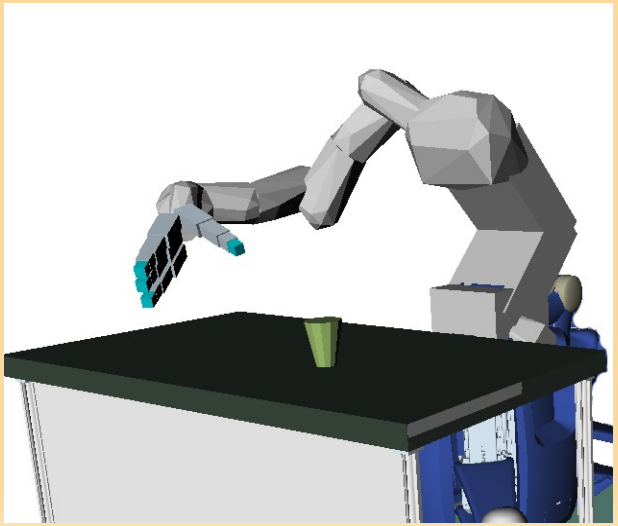


Knowledge representation

- Symbolic Knowledge

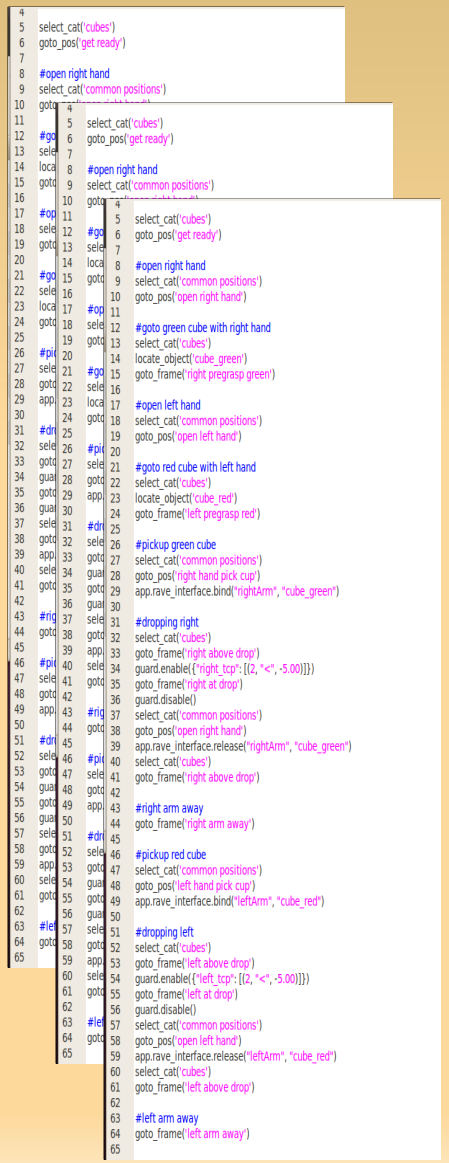
States	Actions
On (Cup1, table) Holding (Teapot, RightHand) Empty (Cup1) ...	Operator: Grab(hand, obj) Preconditions: Free(hand), Clear(obj) Effects: NOT Free(hand), NOT On(obj, table), Holding(hand, obj)

- Geometric Knowledge
  - Geometries of the robot and objects (trimeshes)
  - Position and orientation of objects (transformation matrices)
  - Configuration of the robot (vector)
  - Relative positions (transformation matrices)



Reuse of knowledge: Abstraction

Example programs

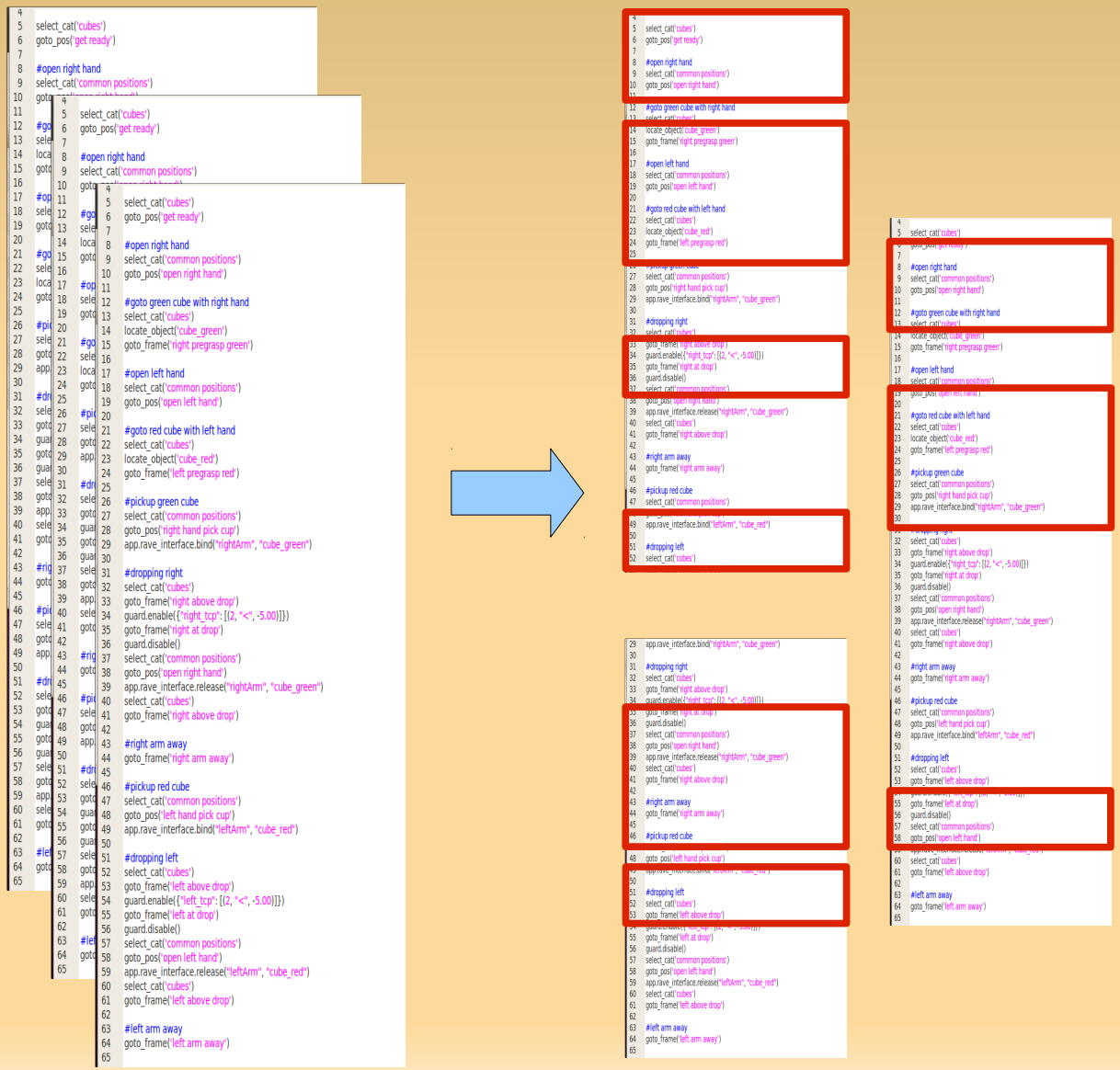


```
select_cat('cubes')
goto_pos('get ready')

#open right hand
select_cat('common positions')
goto_pos('open right hand')

#goto green cube with right hand
select_cat('cubes')
locate_object('cube_green')
goto_frame('right pregrasp green')

#open left hand
select_cat('common positions')
goto_pos('open left hand')

#goto red cube with left hand
select_cat('cubes')
locate_object('cube_red')
goto_frame('left pregrasp red')

#pickup green cube
select_cat('common positions')
goto_pos('right hand pick cup')
app.rave_interface.bind("rightArm", "cube_green")

#dropping right
select_cat('cubes')
goto_frame('right above drop')
guard.enable({"right_tcp": [(2, "<", -5.00)]})
goto_frame('right at drop')
guard.disable()
select_cat('common positions')
goto_pos('open right hand')
app.rave_interface.release("rightArm", "cube_green")
select_cat('cubes')
goto_frame('right above drop')

#right arm away
goto_frame('right arm away')

#pickup red cube
select_cat('common positions')
goto_pos('left hand pick cup')
app.rave_interface.bind("leftArm", "cube_red")

#dropping left
select_cat('cubes')
goto_frame('left above drop')
guard.enable({"left_tcp": [(2, "<", -5.00)]})
goto_frame('left at drop')
guard.disable()
select_cat('common positions')
goto_pos('open left hand')
app.rave_interface.release("leftArm", "cube_red")
select_cat('cubes')
goto_frame('left above drop')

#left arm away
goto_frame('left arm away')
```

Reuse of knowledge: Abstraction

Example programs

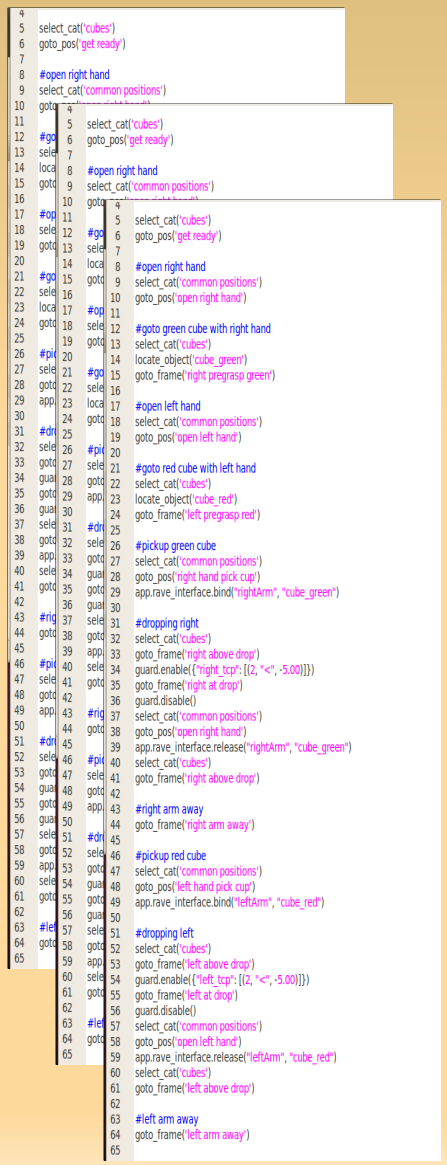
Abstraction / Learning



Reuse of knowledge: Abstraction

Example programs

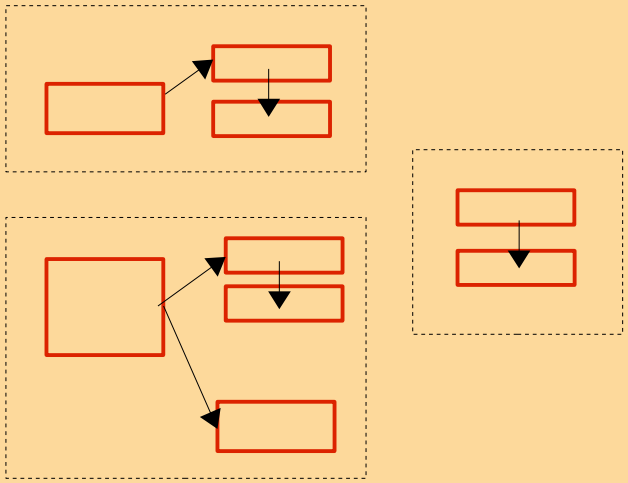
Abstraction / Learning



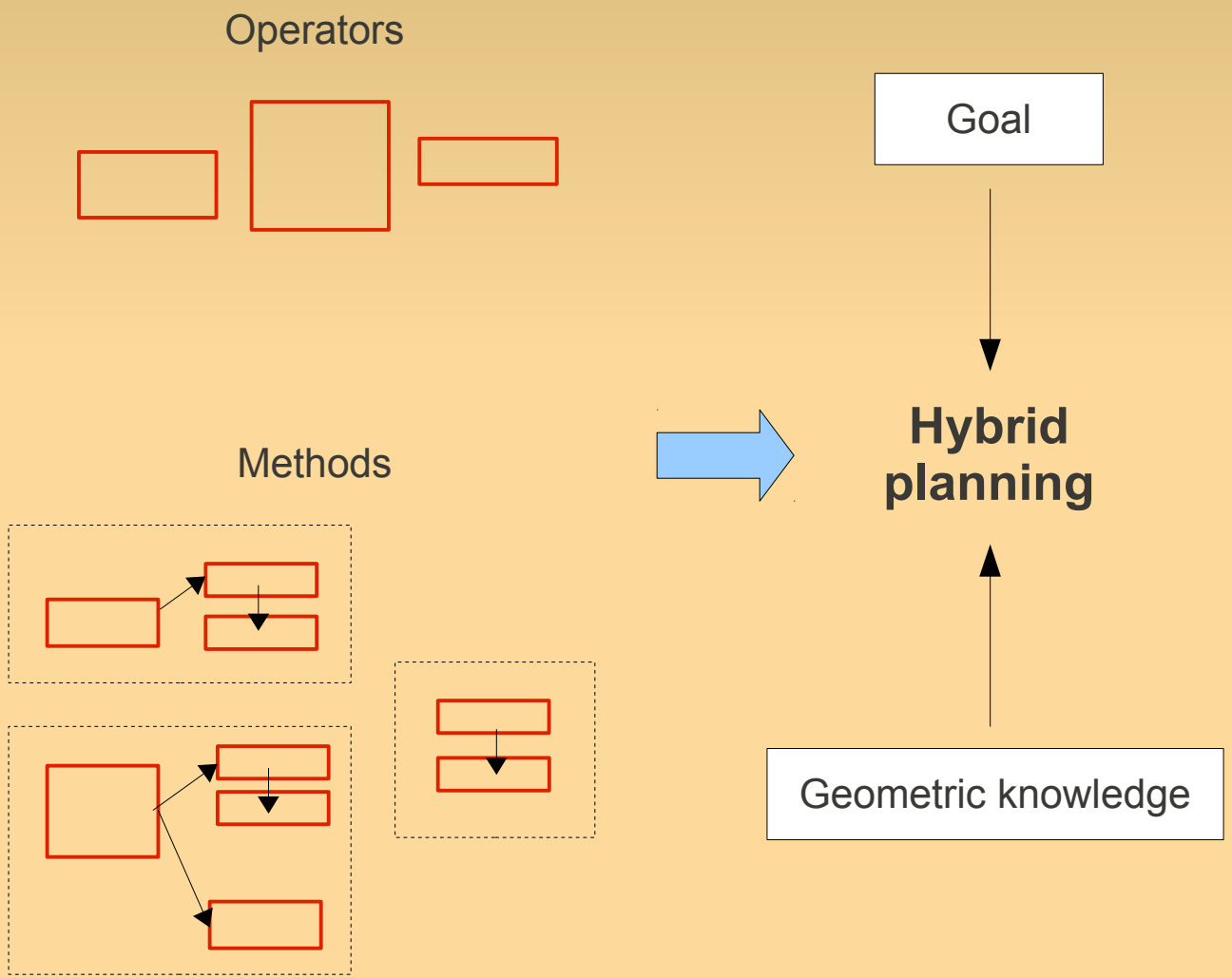
```
select_cat('cubes')
goto_pos('get ready')

#open right hand
select_cat('common positions')
goto_pos('open right hand')

#goto green cube with right hand
select_cat('cubes')
locate_object('cube_green')
goto_frame('right pregrasp green')

#open left hand
select_cat('common positions')
goto_pos('open left hand')

#goto red cube with left hand
select_cat('cubes')
locate_object('cube_red')
goto_frame('left pregrasp red')

#pickup green cube
select_cat('common positions')
goto_pos('right hand pick cup')
app.rave_interface.bind("rightArm", "cube_green")

#dropping right
select_cat('cubes')
goto_frame('right above drop')
guard.enable({"right_tcp": [(2, "<", -5.00)]})
goto_frame('right at drop')
guard.disable()
select_cat('common positions')
goto_pos('open right hand')
app.rave_interface.release("rightArm", "cube_green")
select_cat('cubes')
goto_frame('right above drop')

#right arm away
goto_frame('right arm away')

#pickup red cube
select_cat('common positions')
goto_pos('left hand pick cup')
app.rave_interface.bind("leftArm", "cube_red")

#dropping left
select_cat('cubes')
goto_frame('left above drop')
guard.enable({"left_tcp": [(2, "<", -5.00)]})
goto_frame('left at drop')
guard.disable()
select_cat('common positions')
goto_pos('open left hand')
app.rave_interface.release("leftArm", "cube_red")
select_cat('cubes')
goto_frame('left above drop')

#left arm away
goto_frame('left arm away')
```

Operators

Methods



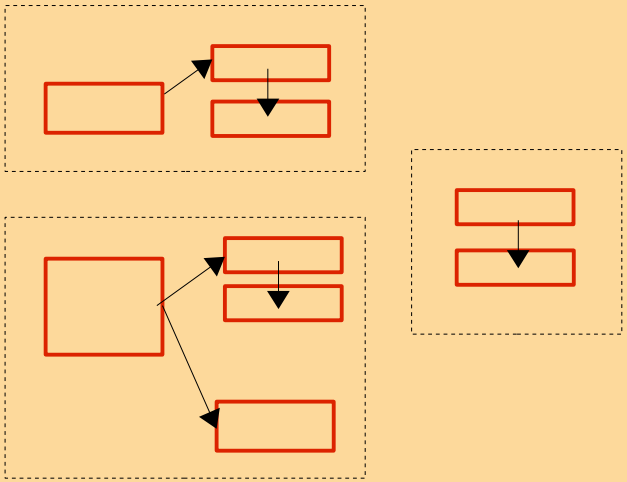
Reuse of knowledge: solving a new task



Reuse of knowledge: solving a new task

Operators

Methods



Goal

Hybrid planning

Geometric knowledge

Python script

```
select_cat('cubes')
goto_pos('get ready')

#open right hand
select_cat('common positions')
goto_pos('open right hand')

#goto green cube with right hand
select_cat('cubes')
locate_object('cube_green')
goto_frame('right pregrasp green')

#open left hand
select_cat('common positions')
goto_pos('open left hand')

#goto red cube with left hand
select_cat('cubes')
locate_object('cube_red')
goto_frame('left pregrasp red')

#pickup green cube
select_cat('common positions')
goto_pos('right hand pick cup')
app.rave_interface.bind("rightArm", "cube_green")

#dropping right
select_cat('cubes')
goto_frame('right above drop')
guard.enable({"right_tcp": [(2, "<", -5.00)]})
goto_frame('right at drop')
guard.disable()
select_cat('common positions')
goto_pos('open right hand')
app.rave_interface.release("rightArm", "cube_green")
select_cat('cubes')
goto_frame('right above drop')

#right arm away
goto_frame('right arm away')

#pickup red cube
select_cat('common positions')
goto_pos('left hand pick cup')
app.rave_interface.bind("leftArm", "cube_red")

#dropping left
select_cat('cubes')
goto_frame('left above drop')
guard.enable({"left_tcp": [(2, "<", -5.00)]})
goto_frame('left at drop')
guard.disable()
select_cat('common positions')
goto_pos('open left hand')
app.rave_interface.release("leftArm", "cube_red")
select_cat('cubes')
goto_frame('left above drop')

#left arm away
goto_frame('left arm away')
```

The End

Thanks for your attention.

Questions?