Reactive Task and Motion Planning under Temporal Logic Specifications

Shen Li*, Daehyung Park*, Yoonchang Sung*,

Julie A. Shah, Nicholas Roy

Massachusetts Institute of Technology

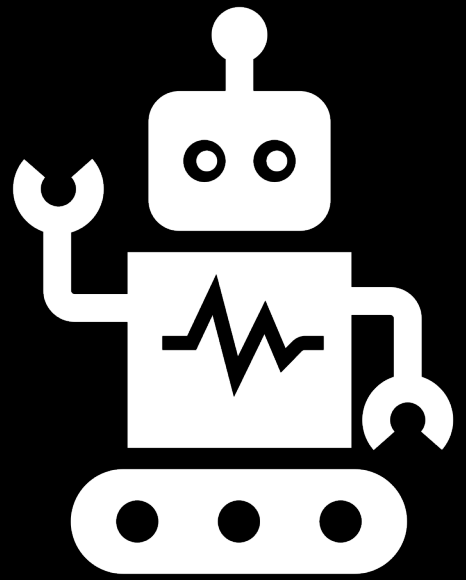
* These authors contributed equally to this work.

Robust Robotics Group

RIRO LAB
Robust Intelligence & Robotics Lab

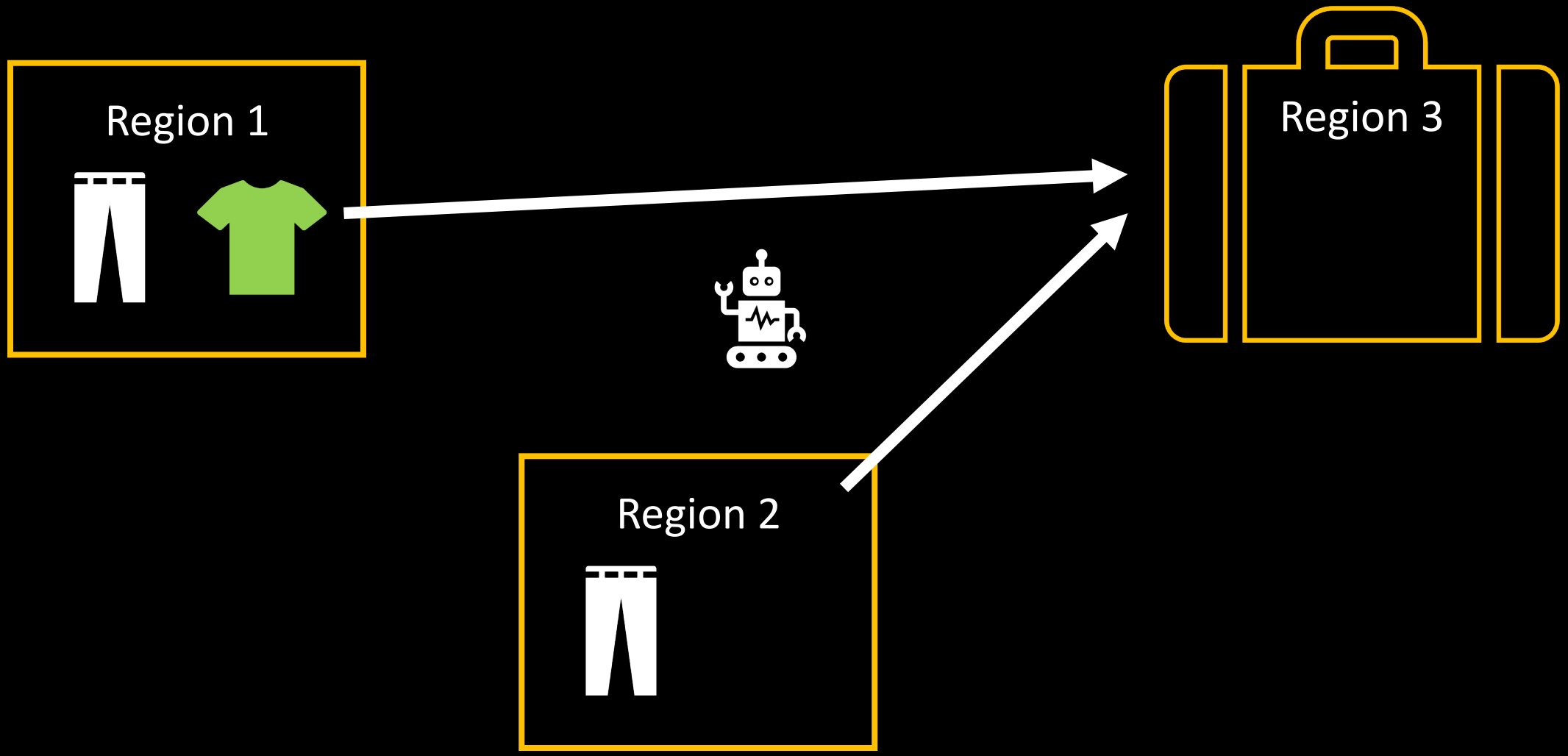
Learning & Intelligent Systems

Robot-assisted packing

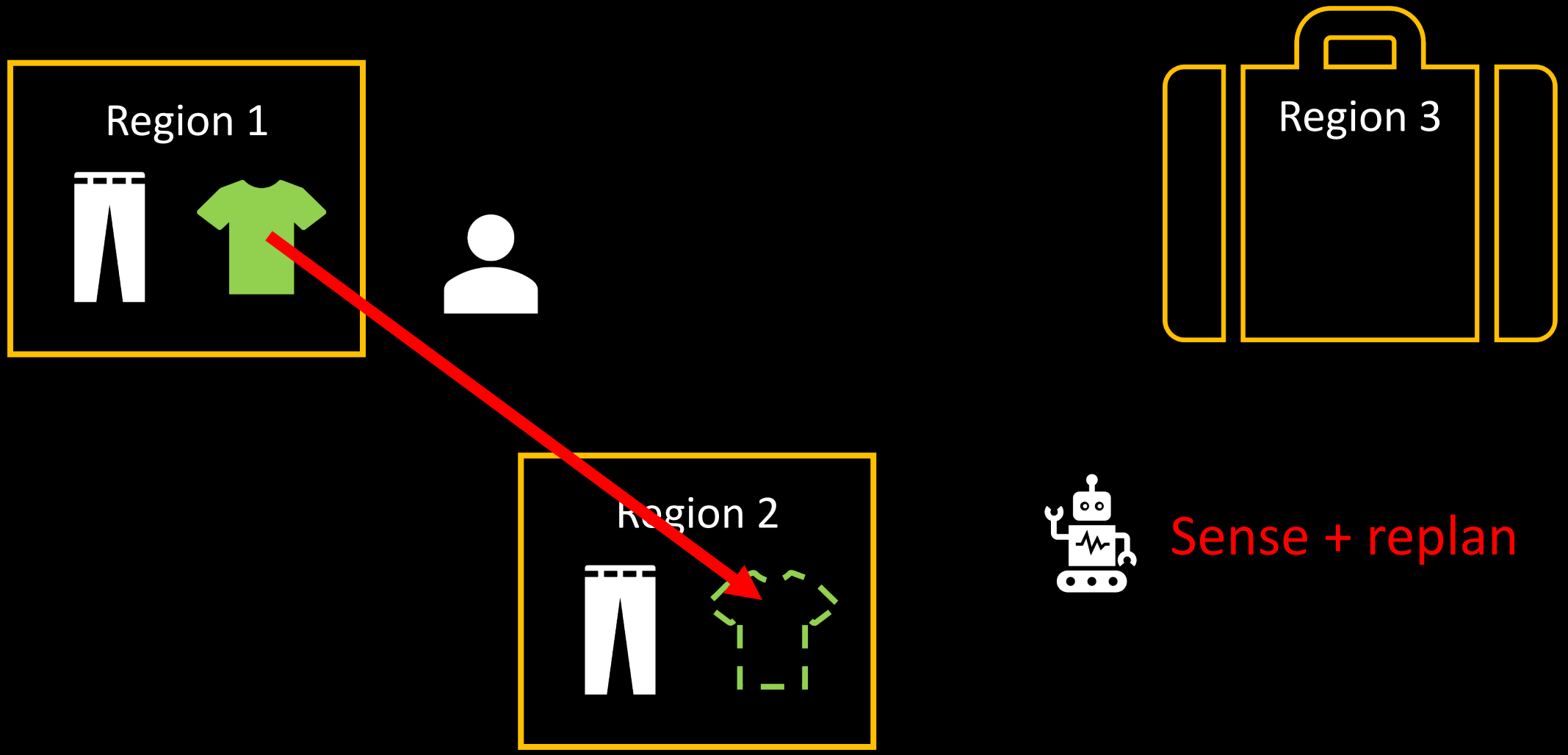


https://www.roadaffair.com/best-travel-organizers/

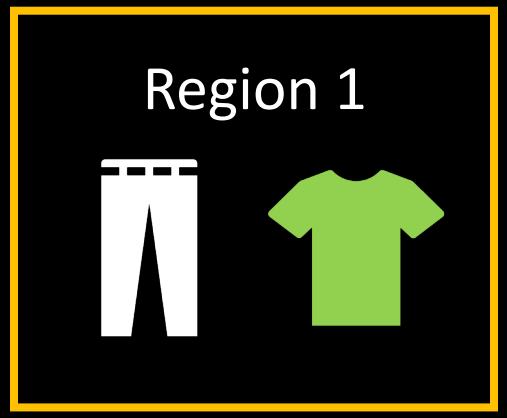
Task-and-motion planning

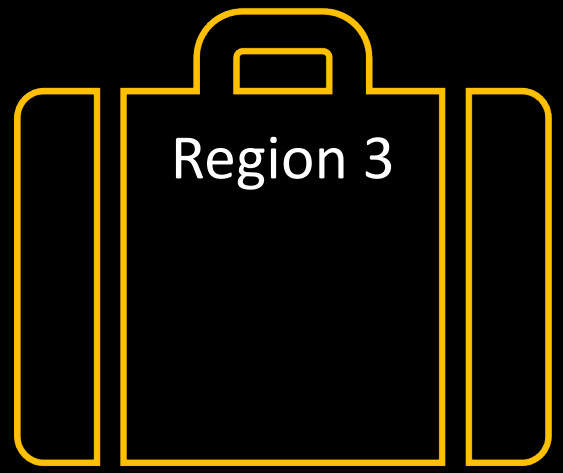


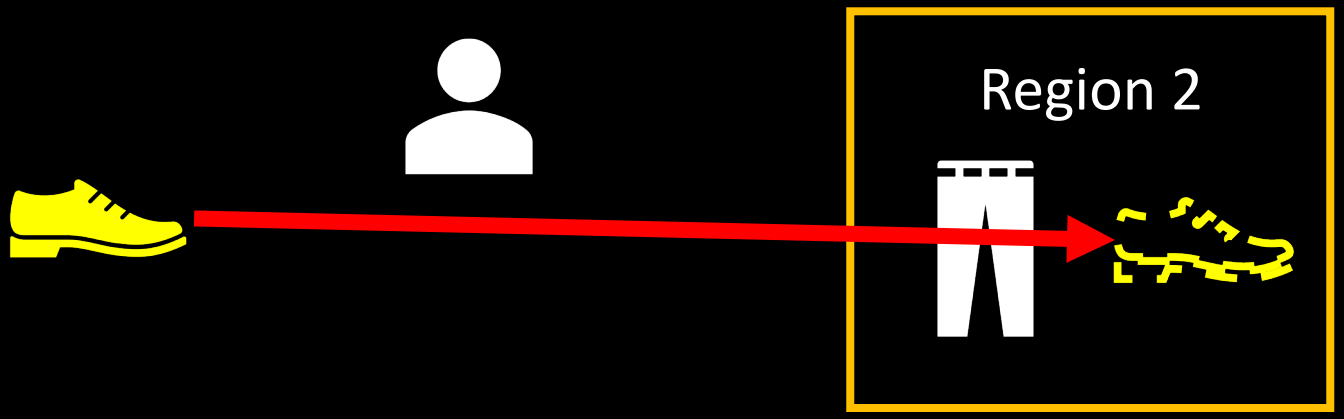
Human can relocate objects

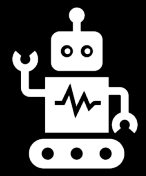


Human can add objects



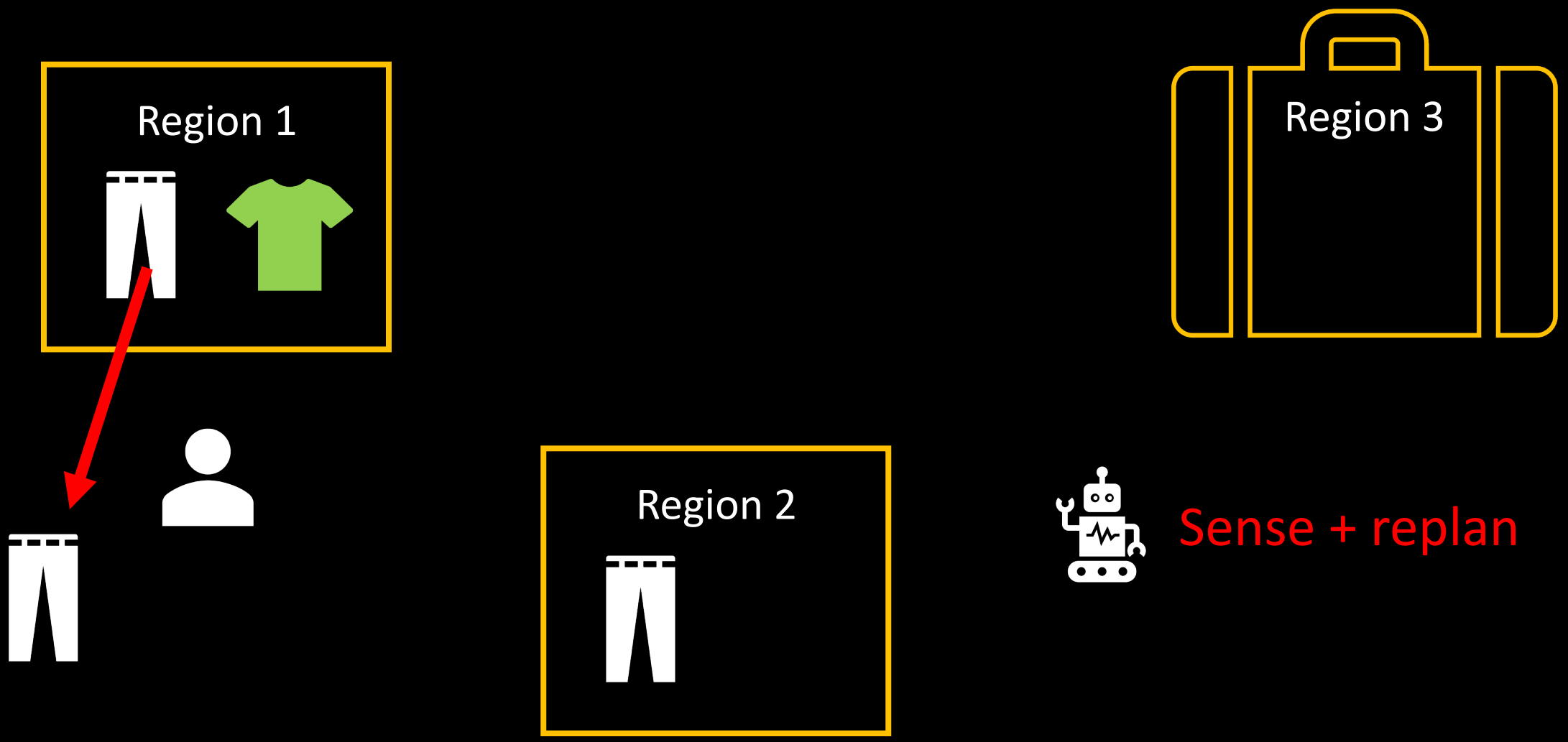




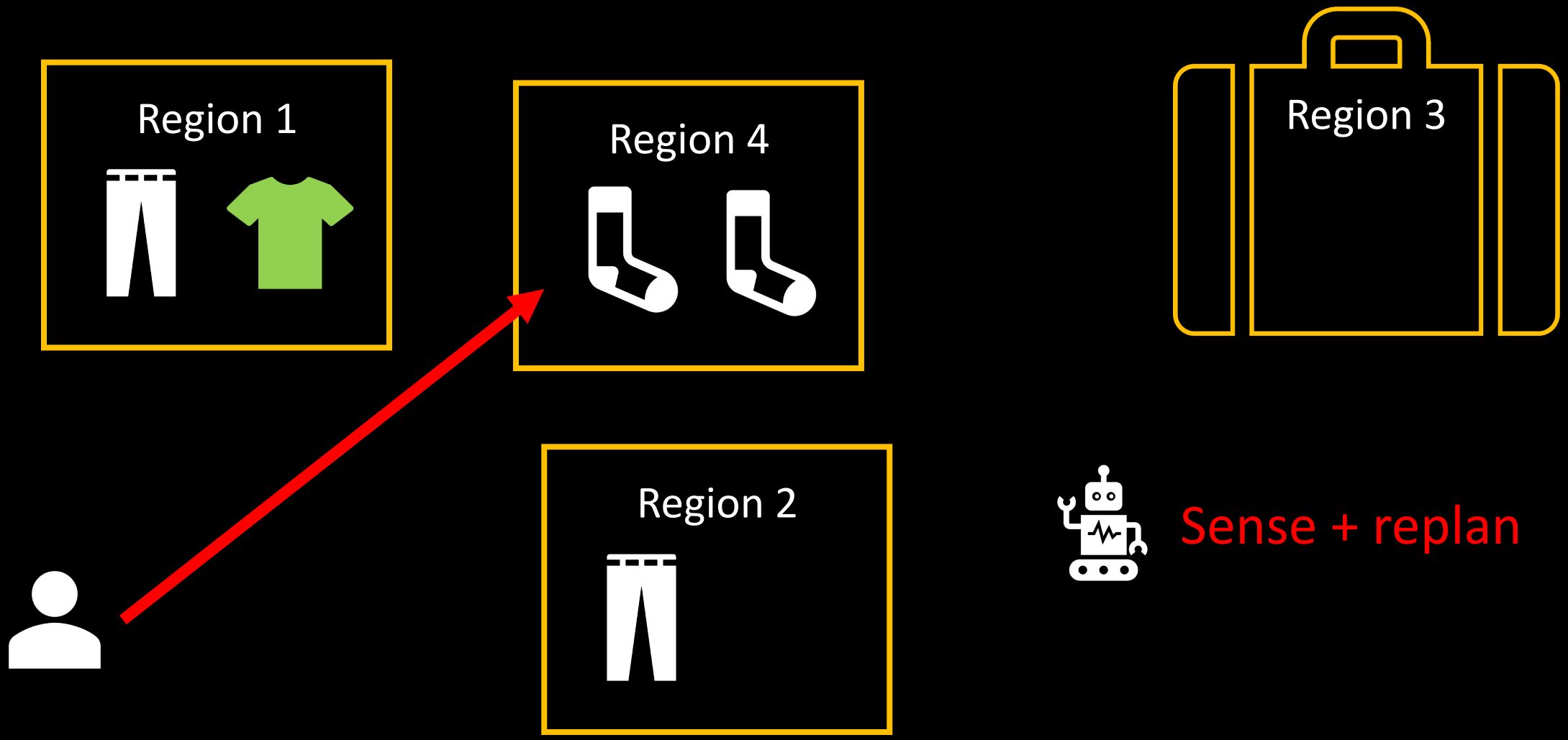


Sense + replan

Human can remove objects



Human can add regions



Problem definition

- Task and motion planning
- Environmental changes
  - Unexpected changes made by humans in human-robot interaction

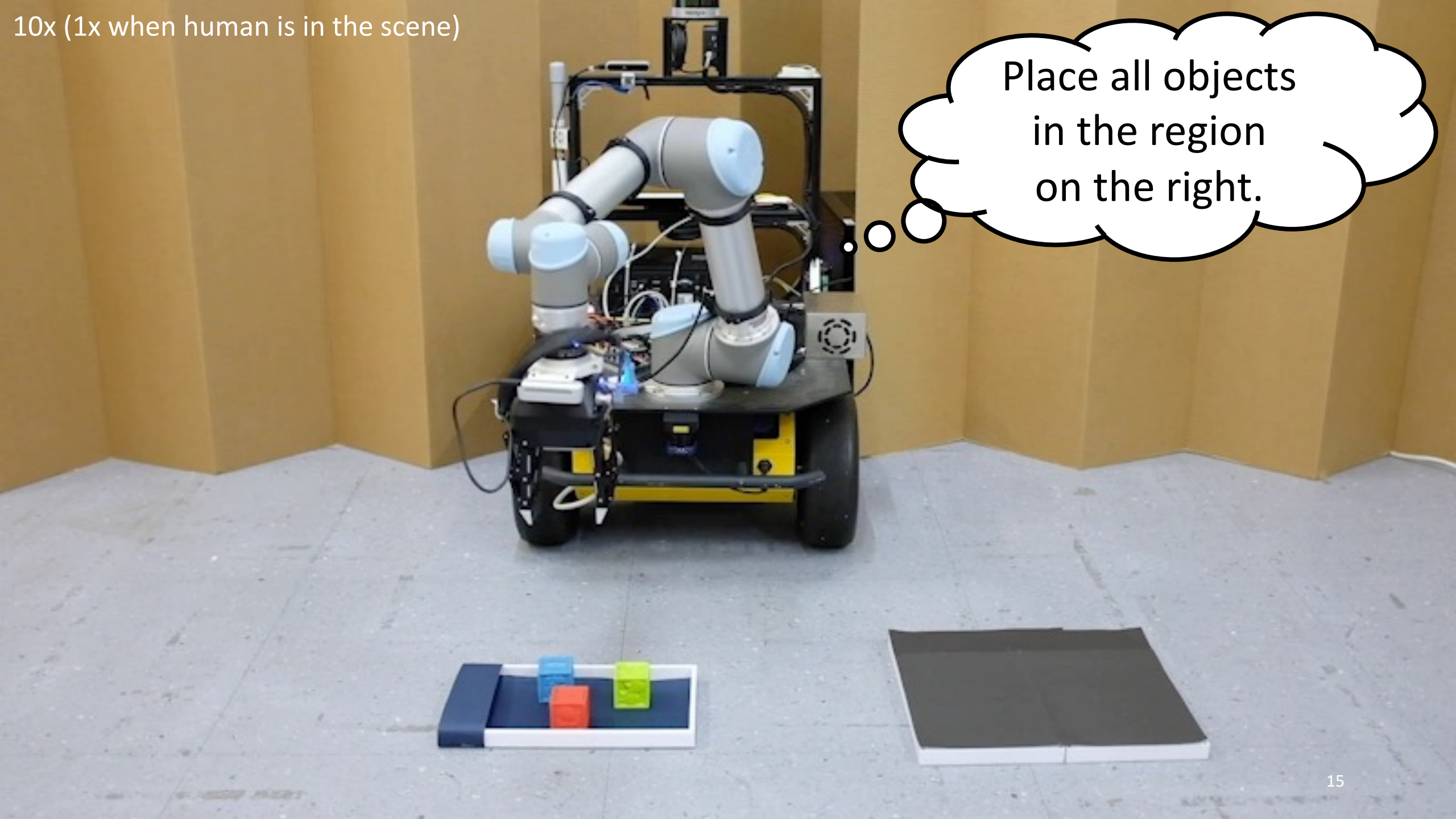
Environmental changes

- Object relocation
- Object addition and removal
- Region addition

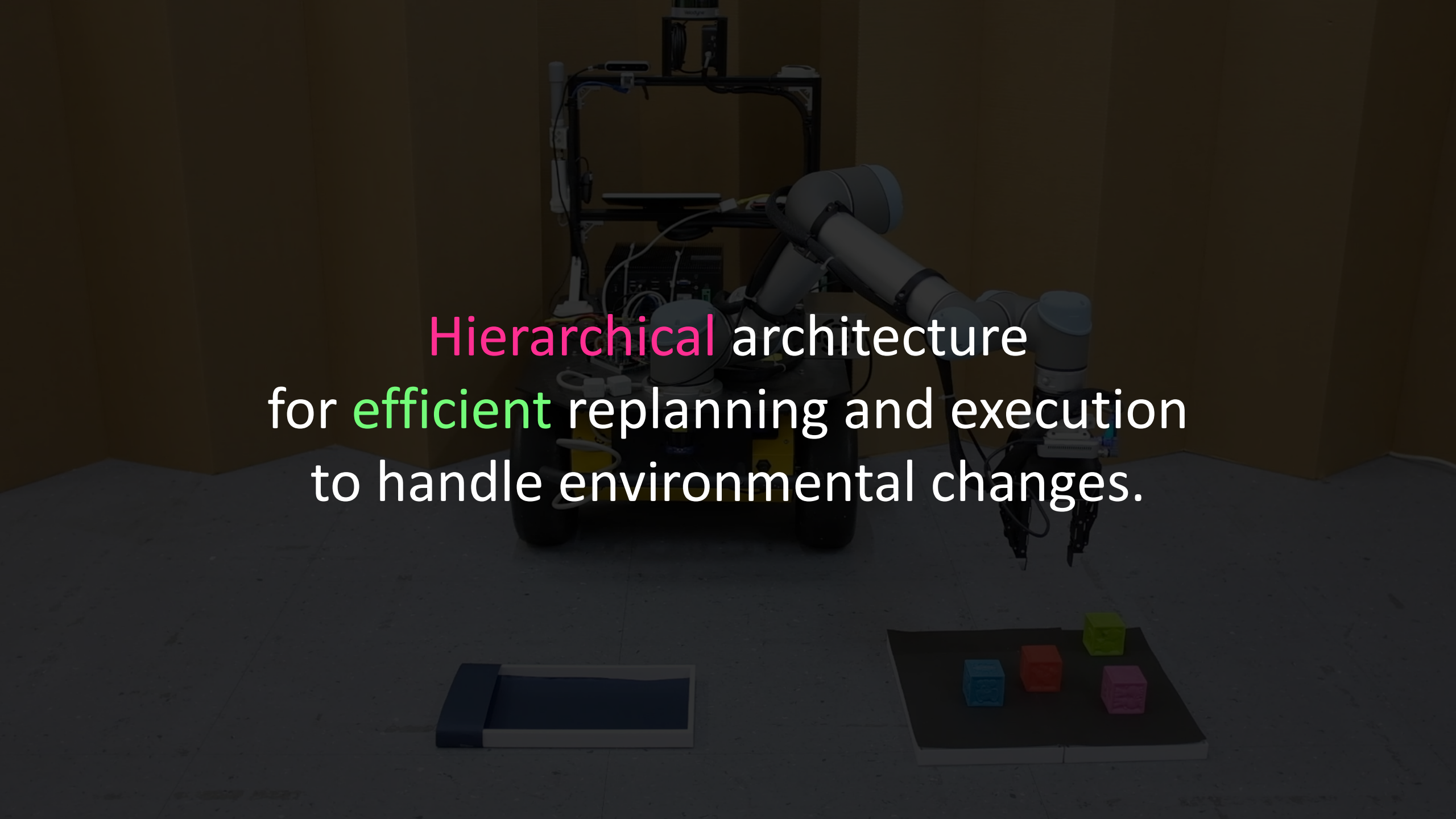
Environmental changes

- Object relocation
- Object addition and removal
- Region addition

Computationally expensive to sense and replan on the fly.


10x (1x when human is in the scene)
Place all objects
in the region
on the right.

Replanning on the fly
is computationally expensive.

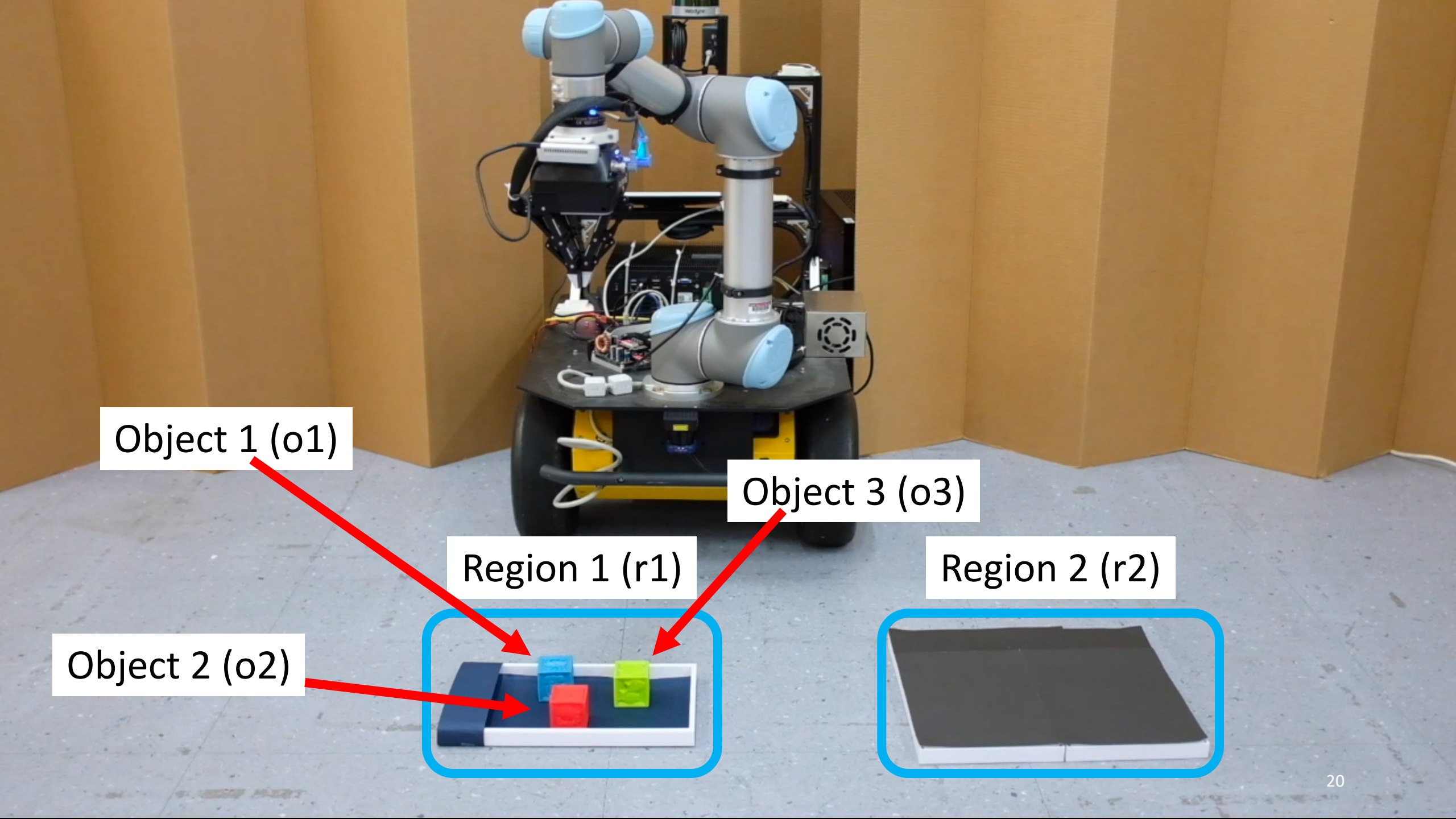

Hierarchical architecture
for efficient replanning and execution
to handle environmental changes.

Hierarchical System Design

To generate reactive behavior

Region 1 (r1)

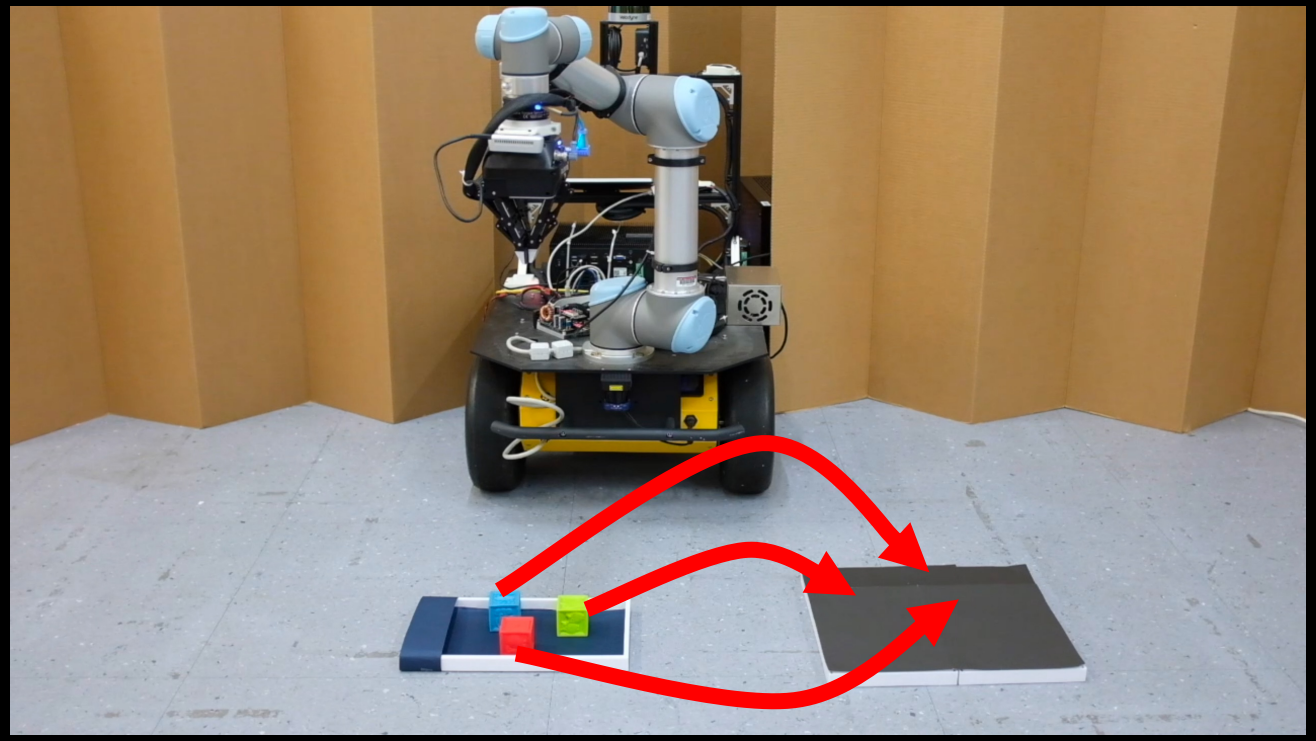
Region 2 (r2)


Object 1 (o1)
Object 3 (o3)
Region 1 (r1)
Region 2 (r2)
Object 2 (o2)

Linear temporal logic (LTL) specification (known)

Eventually always keep all objects in region 2.

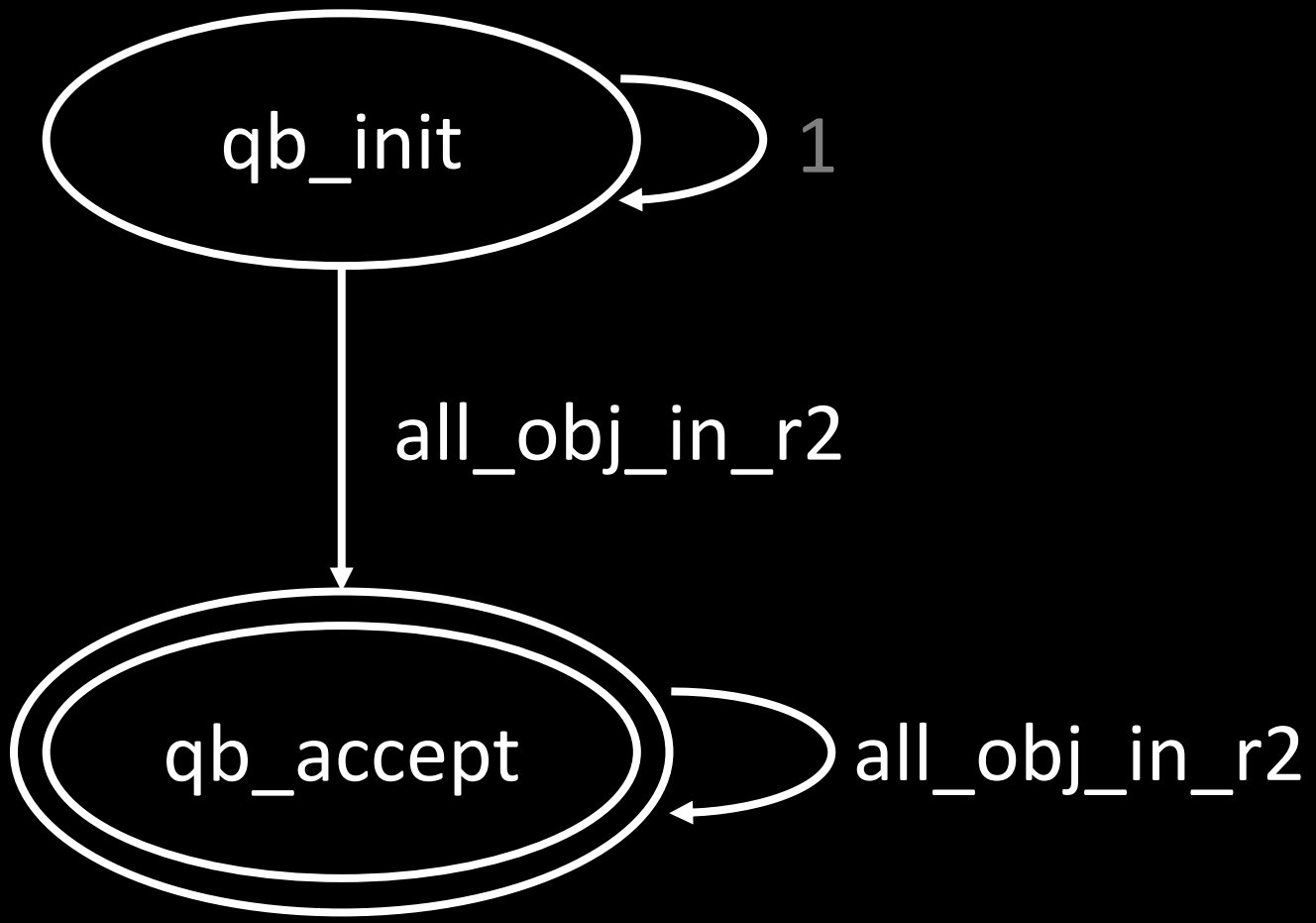
$$\mathcal{FG}(all\_obj\_in\_r_2)$$



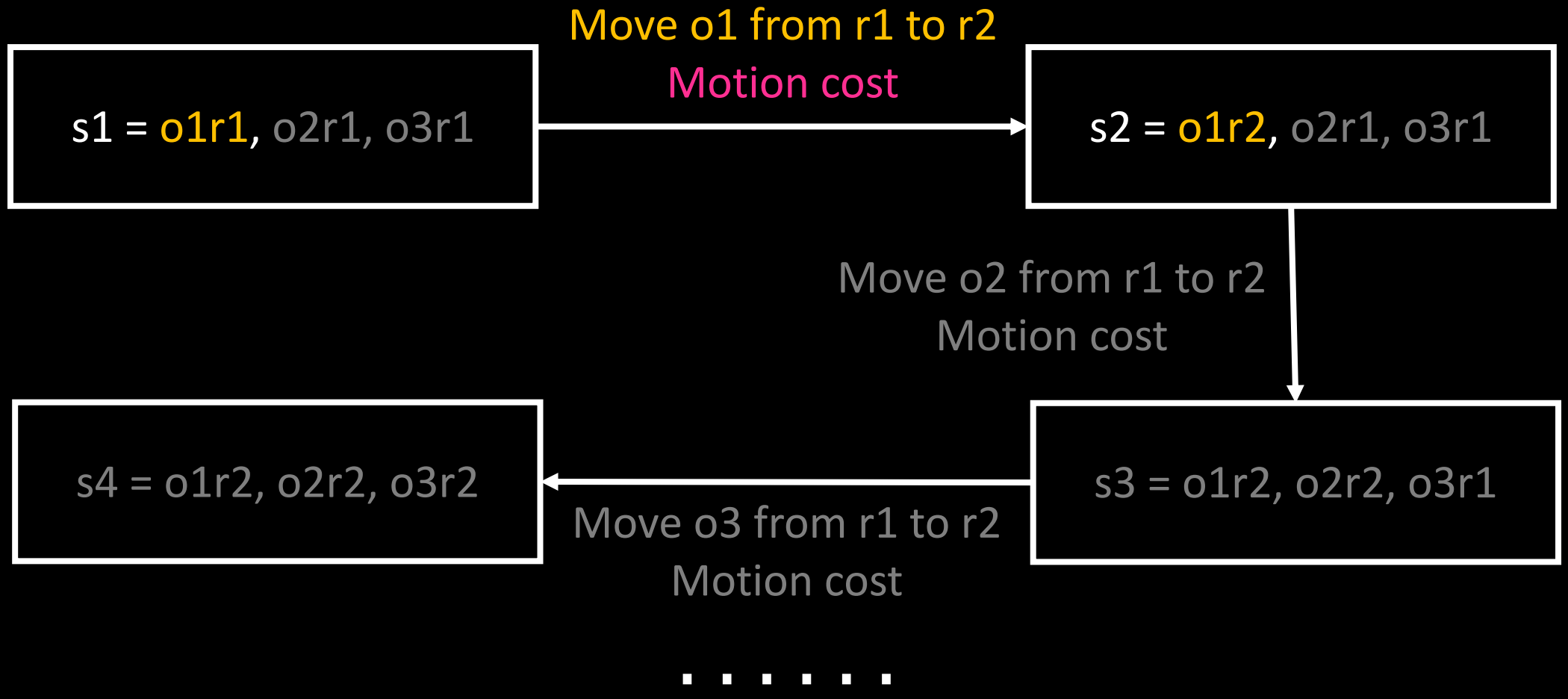
LTL specification → Buchi automaton (BA)

(known)

$$\mathcal{FG}(all\_obj\_in\_r_2)$$



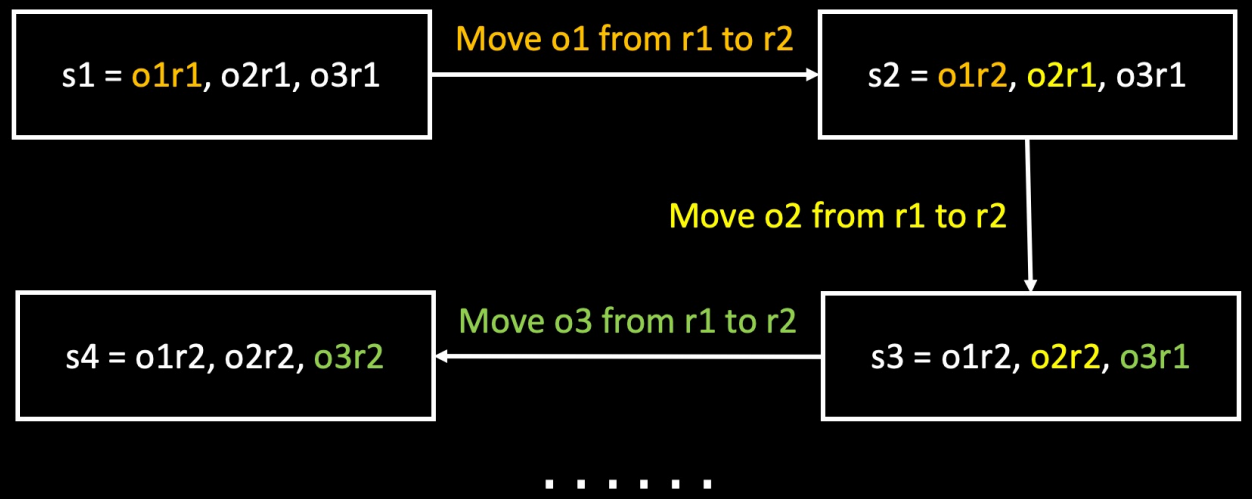
Robot, workspace → Transition system (TS)



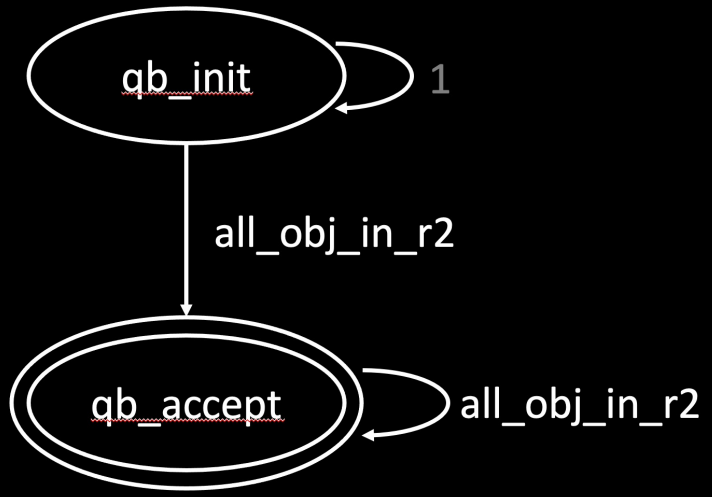
Robot, workspace → Transition system (TS)

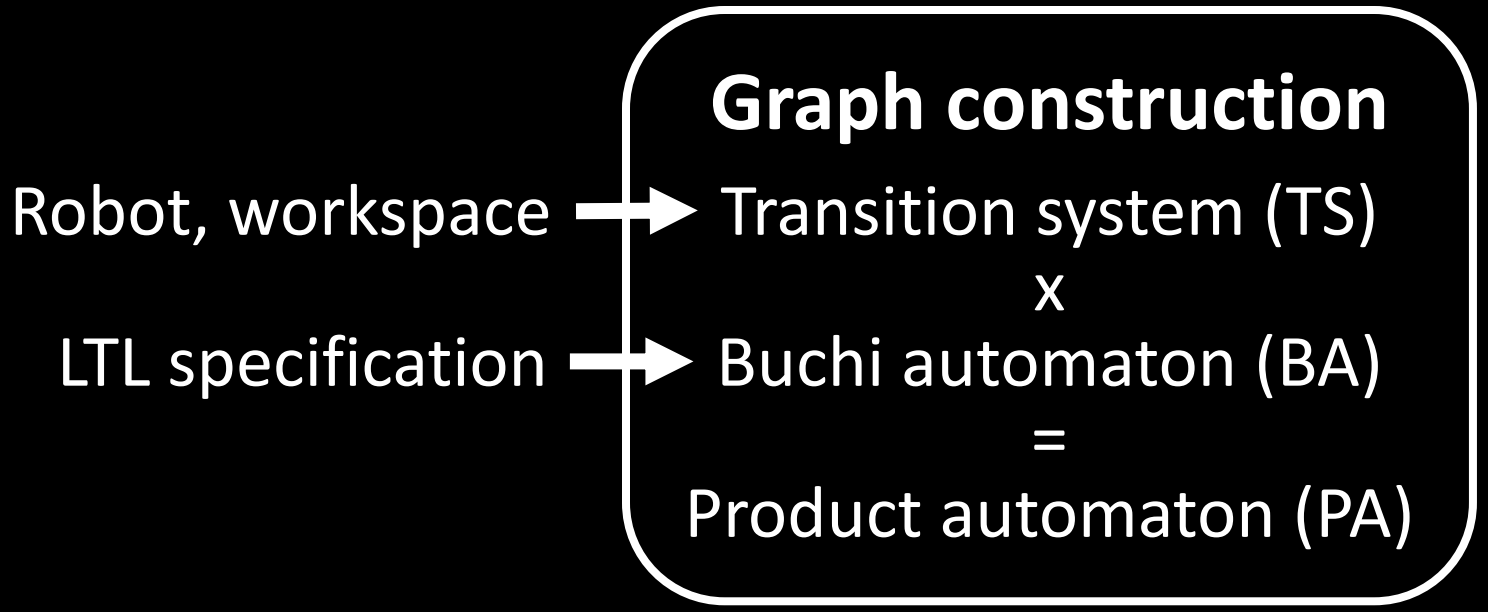
X

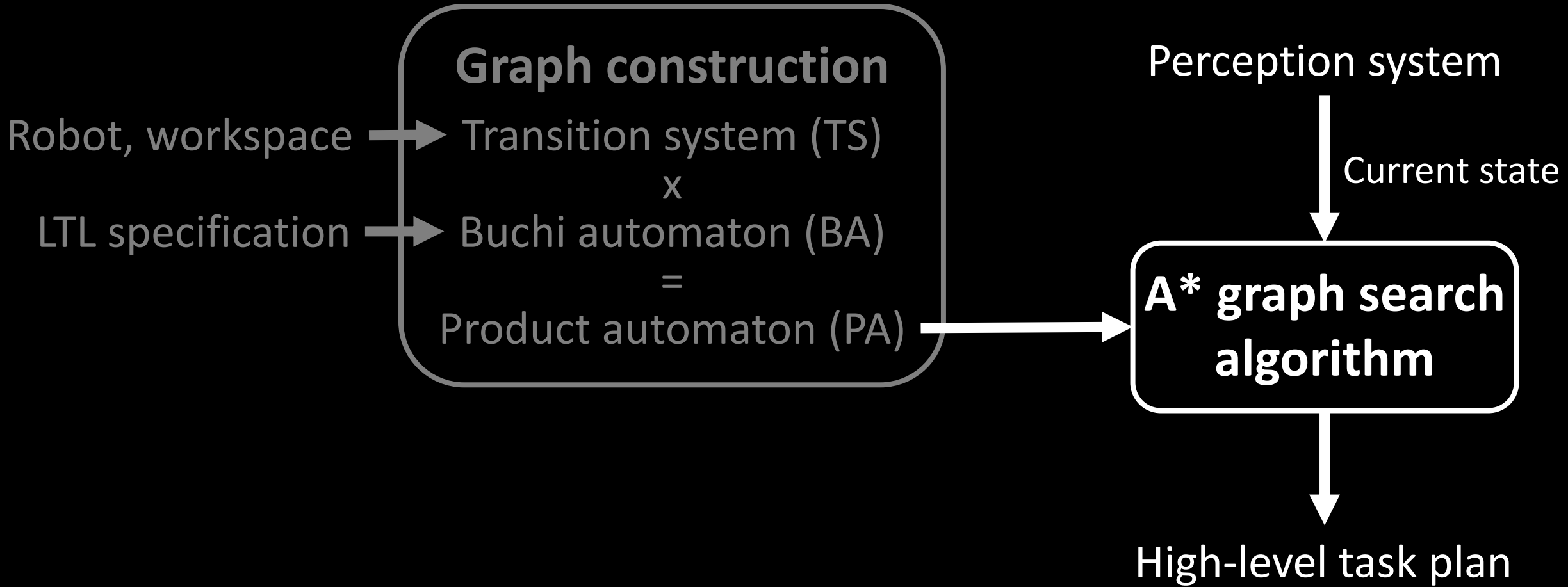
LTL specification → Buchi automaton (BA)



X




Graph construction
Robot, workspace
LTL specification
Transition system (TS)
x
Buchi automaton (BA)
=
Product automaton (PA)


Graph construction
Robot, workspace
LTL specification
Transition system (TS)
x
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search algorithm
High-level task plan

Graph construction

Robot, workspace → Transition system (TS)

x

LTL specification → Buchi automaton (BA)

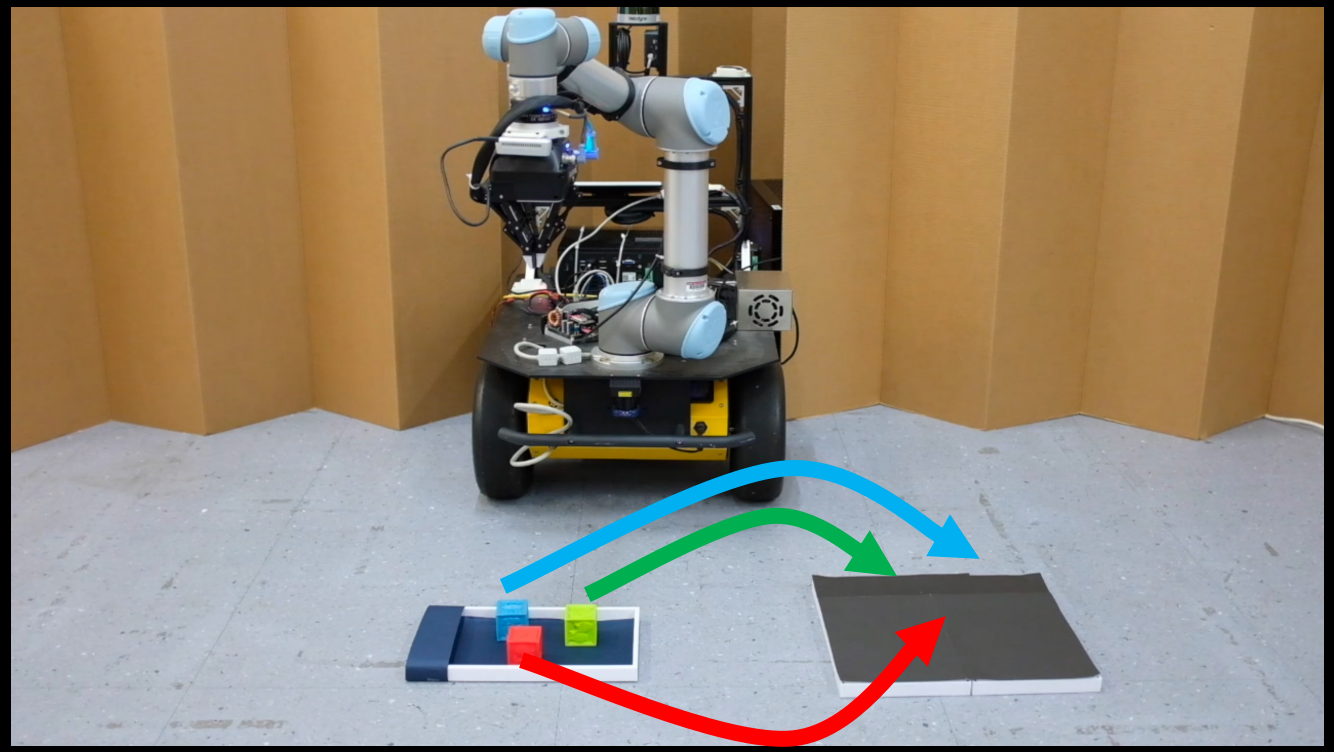
=

Product automaton (PA)

Perception system

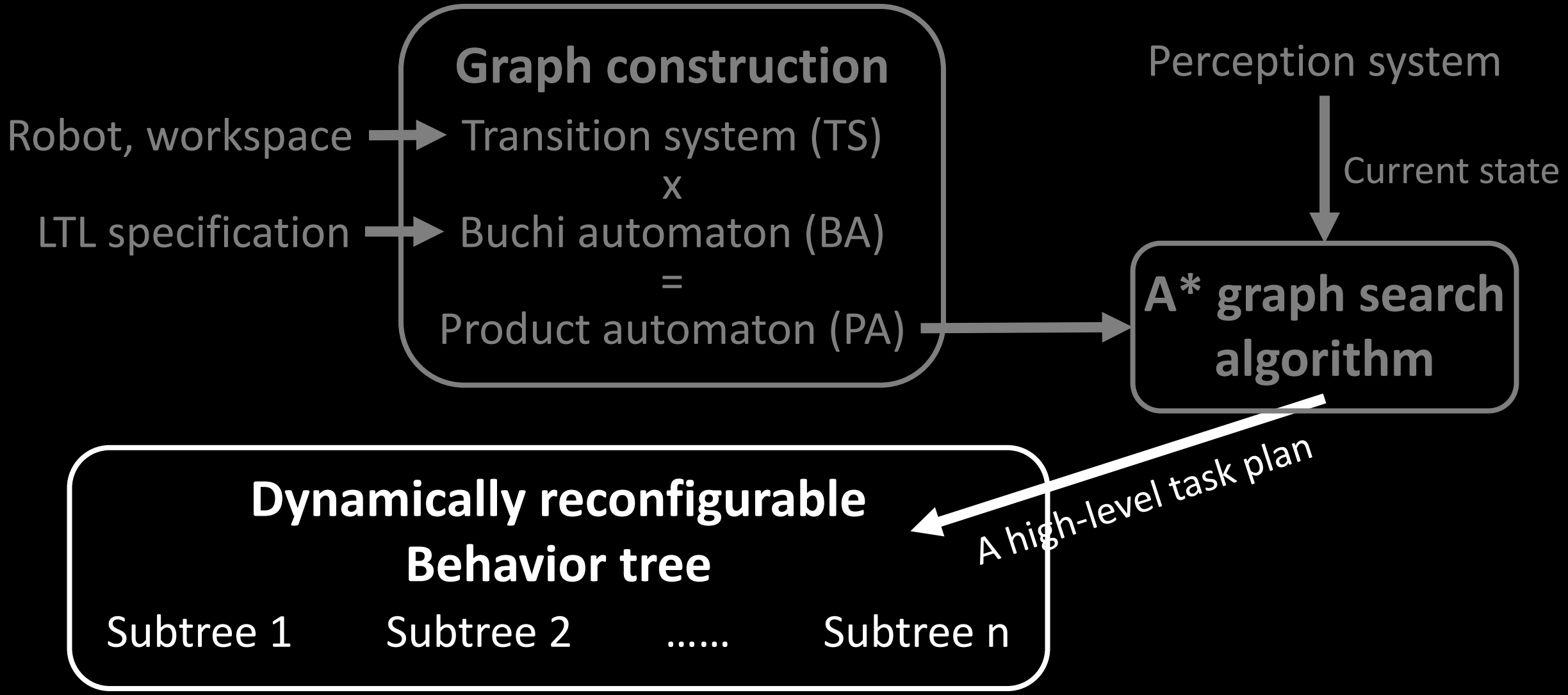
Current state

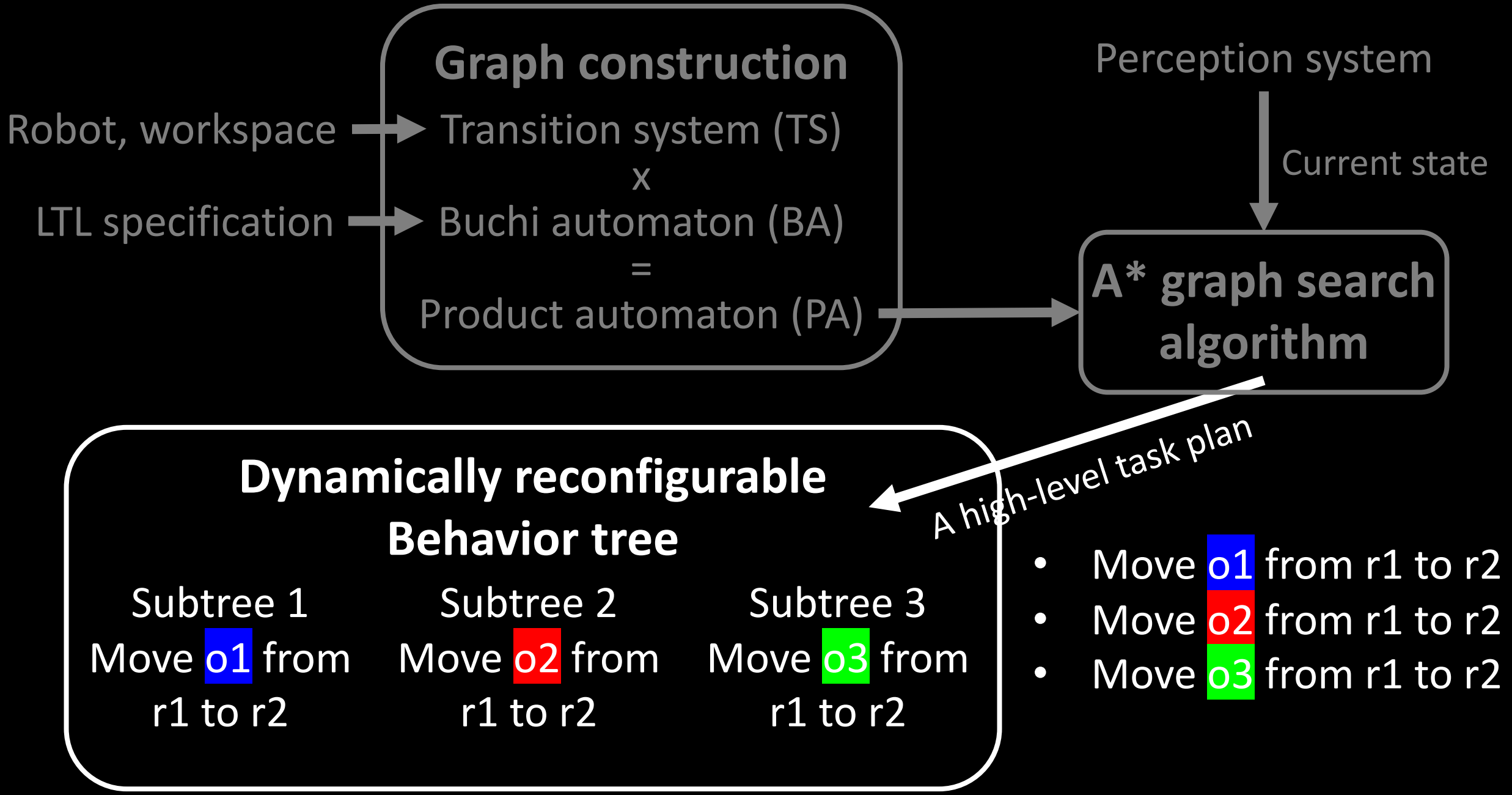
A* graph search algorithm

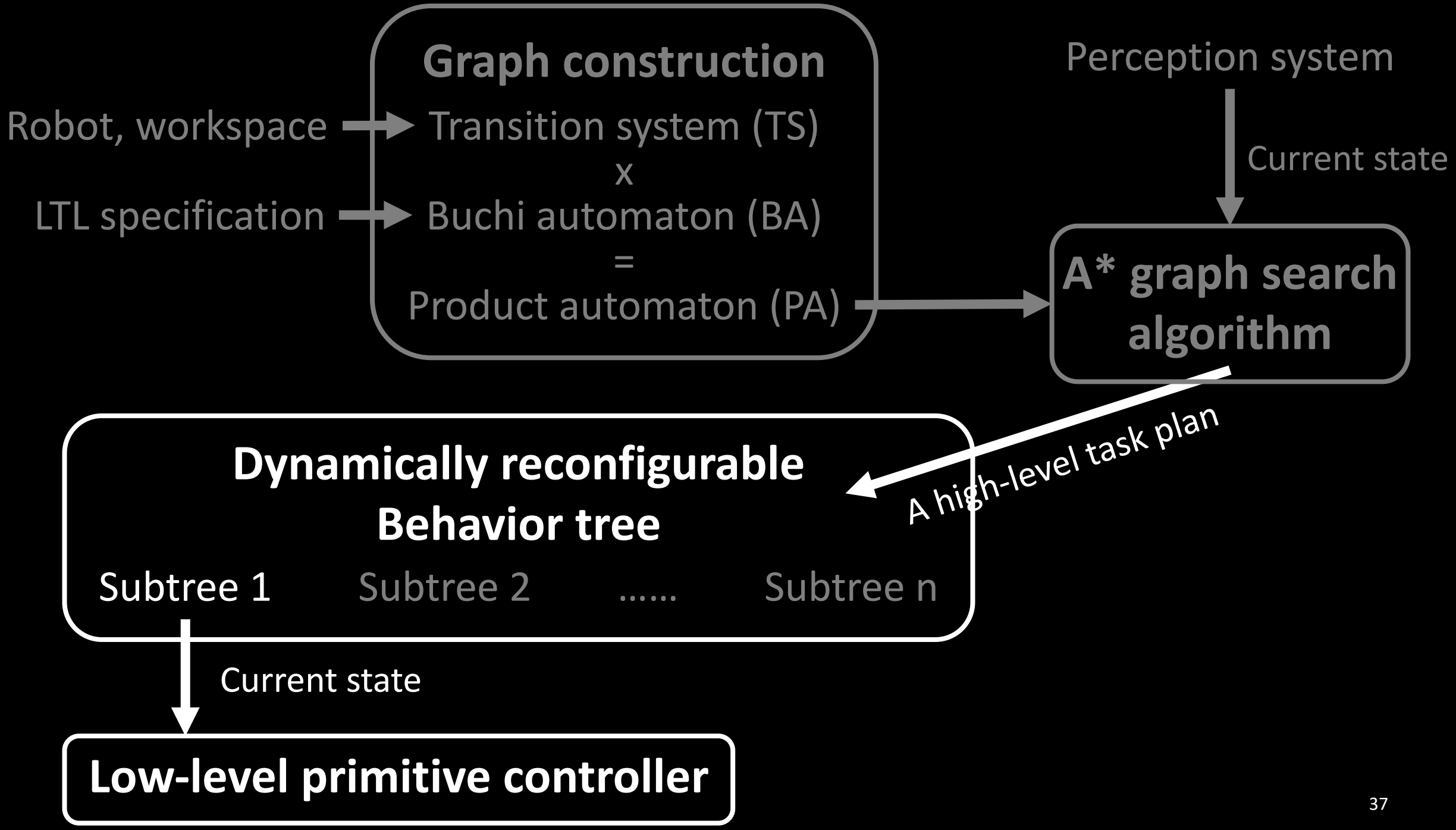


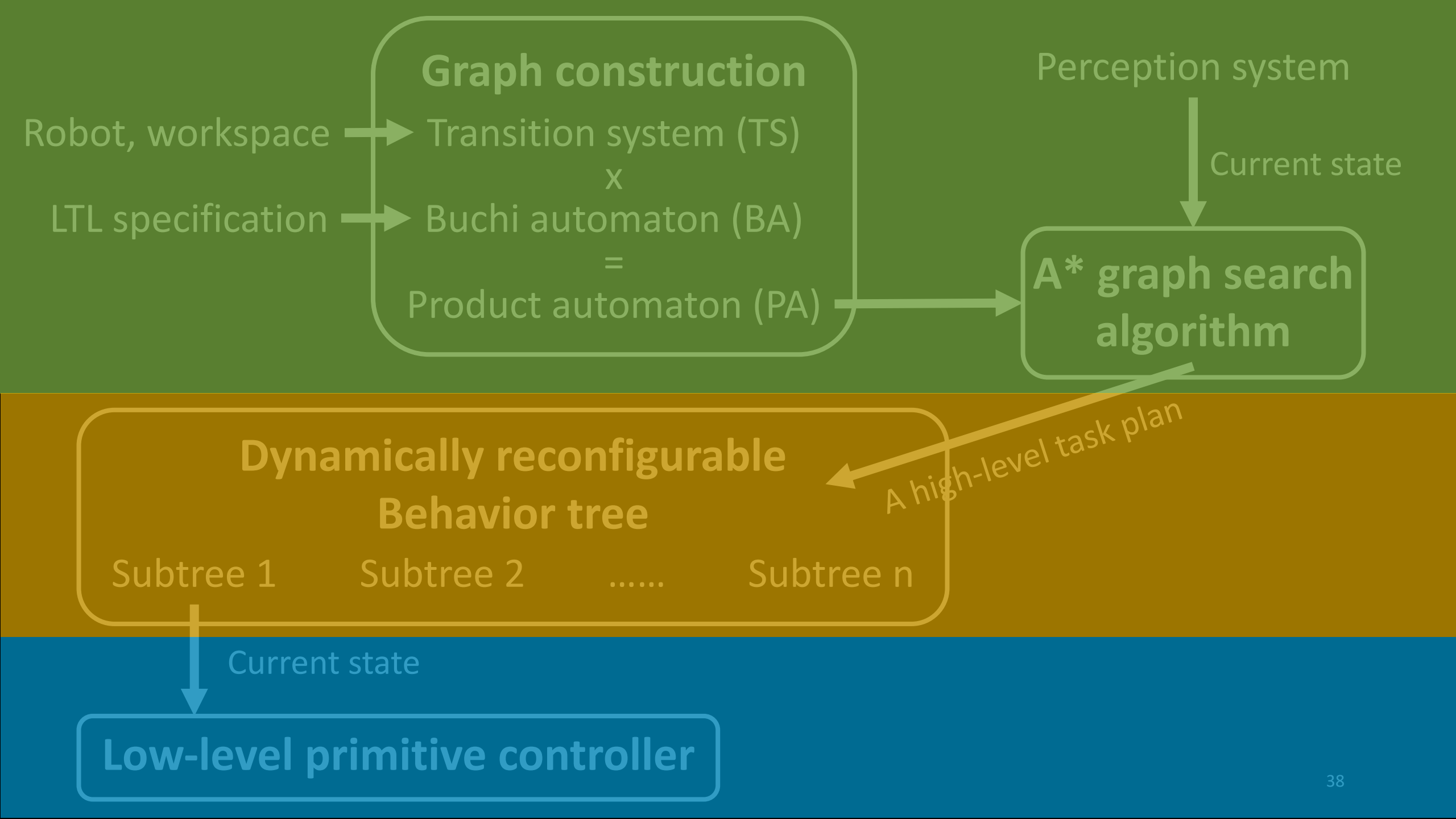
High-level task plan

- Move o1 from r1 to r2
- Move o2 from r1 to r2
- Move o3 from r1 to r2


Graph construction
Robot, workspace
Transition system (TS)
x
LTL specification
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search algorithm
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n


Graph construction
Robot, workspace
LTL specification
Transition system (TS)
x
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search algorithm
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Move o1 from r1 to r2
Subtree 2
Move o2 from r1 to r2
Subtree 3
Move o3 from r1 to r2
• Move o1 from r1 to r2
• Move o2 from r1 to r2
• Move o3 from r1 to r2

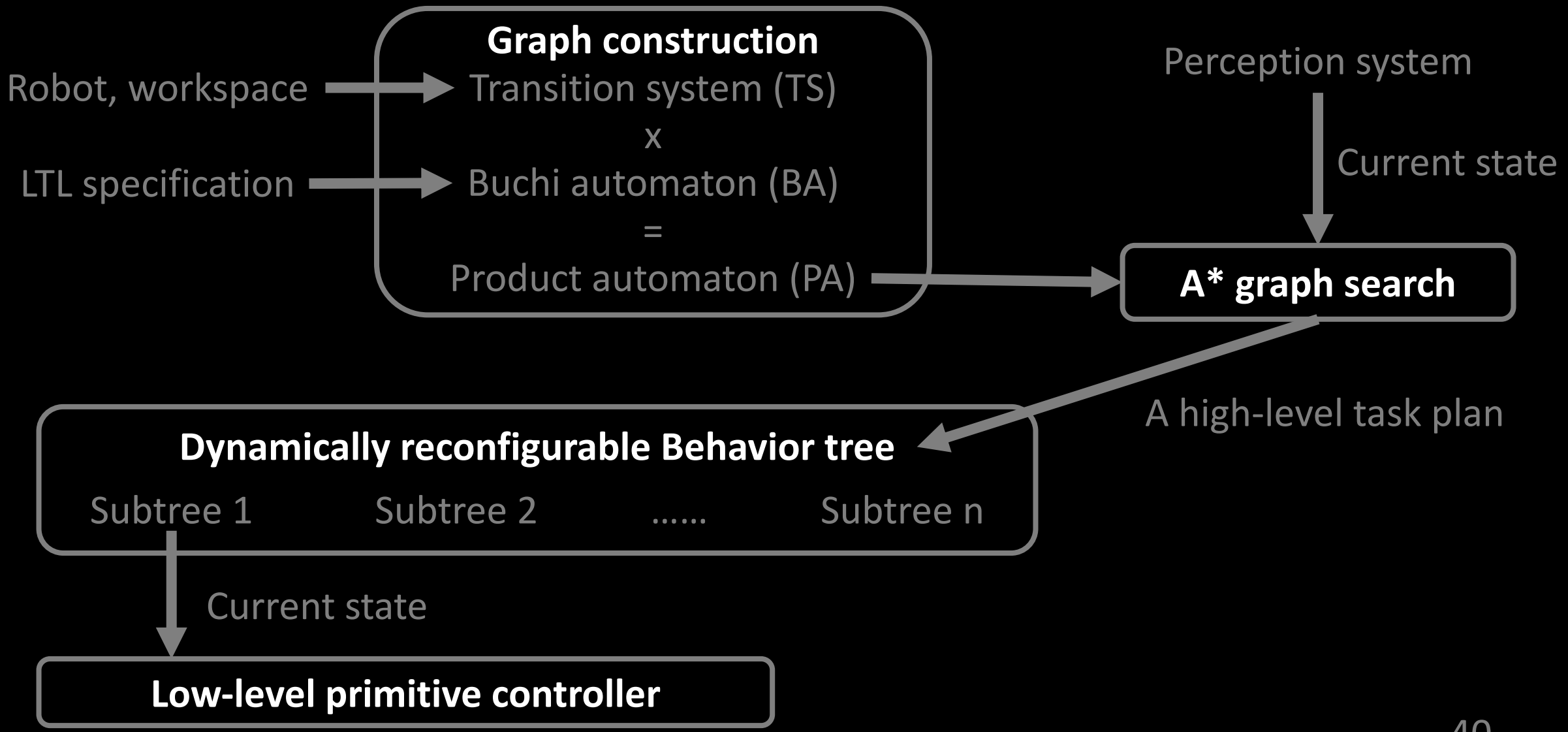

Graph construction
Robot, workspace
Transition system (TS)
x
LTL specification
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search algorithm
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller


Graph construction
Robot, workspace
Transition system (TS)
x
LTL specification
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search algorithm
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller

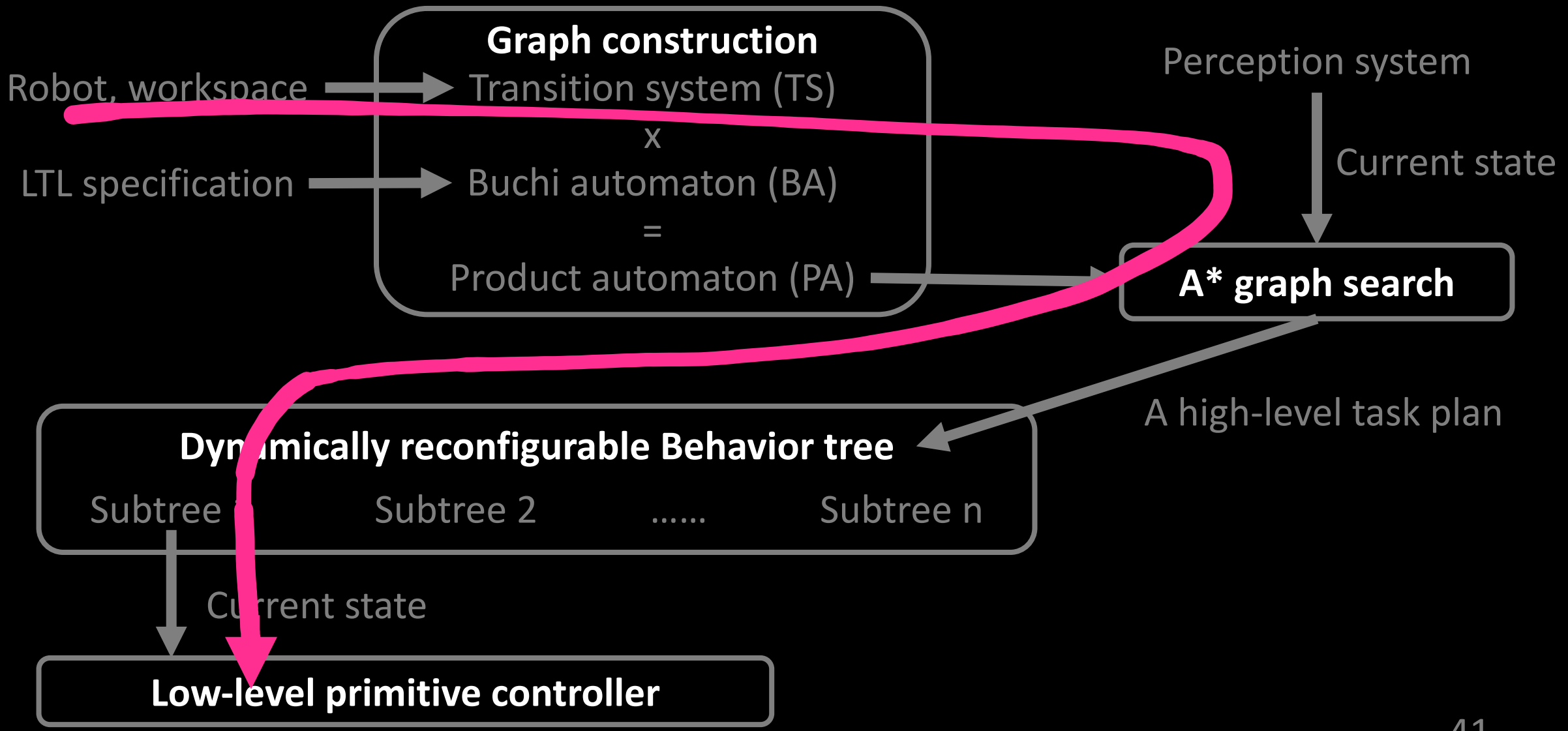
Efficiently handle environmental changes

1. Hierarchical system design
2. Algorithmic design

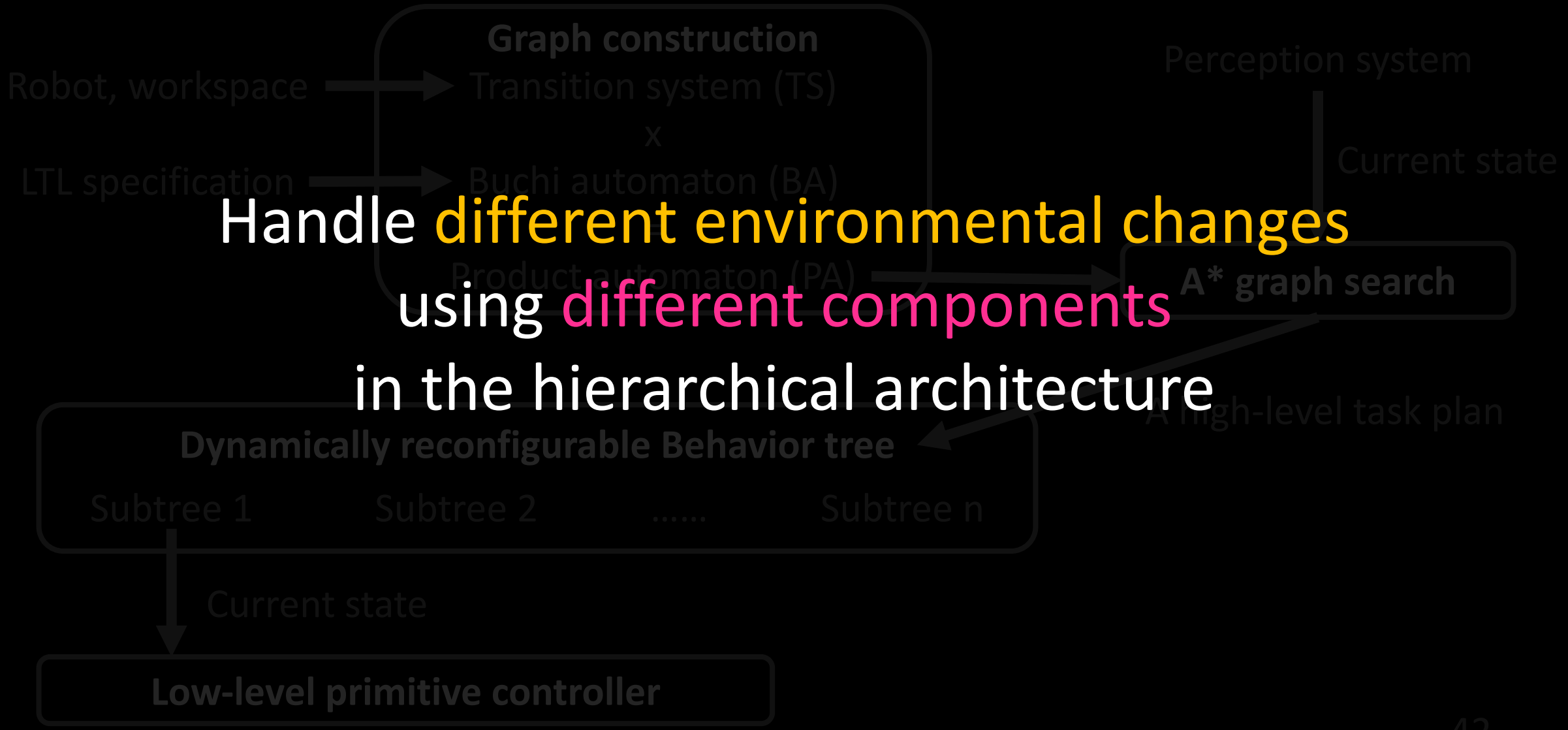
Hierarchical system design to efficiently handle environmental changes



Hierarchical system design to efficiently handle environmental changes



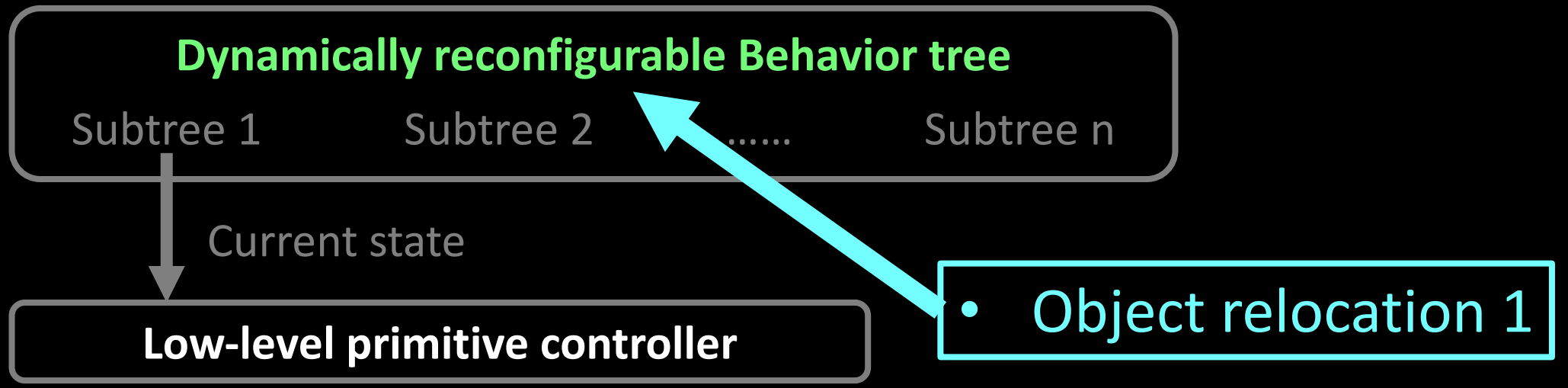
Hierarchical System Design for Efficient Interference Handling



Handle different environmental changes
using different components
in the hierarchical architecture

Environmental changes

- Object relocation
- Object addition and removal
- Region addition

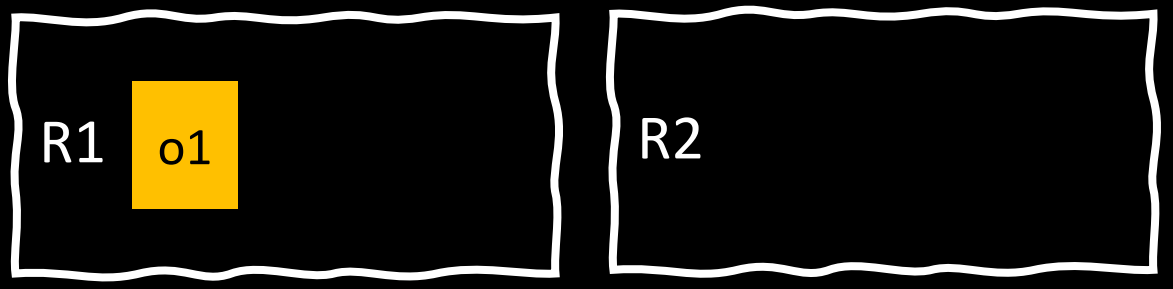

Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller
• Object relocation 1

Behavior tree

Subtree 1:
Move o1 from r1 to r2

Subtree 2:
Move o2 from r1 to r2

Subtree 3:
Move o3 from r1 to r2

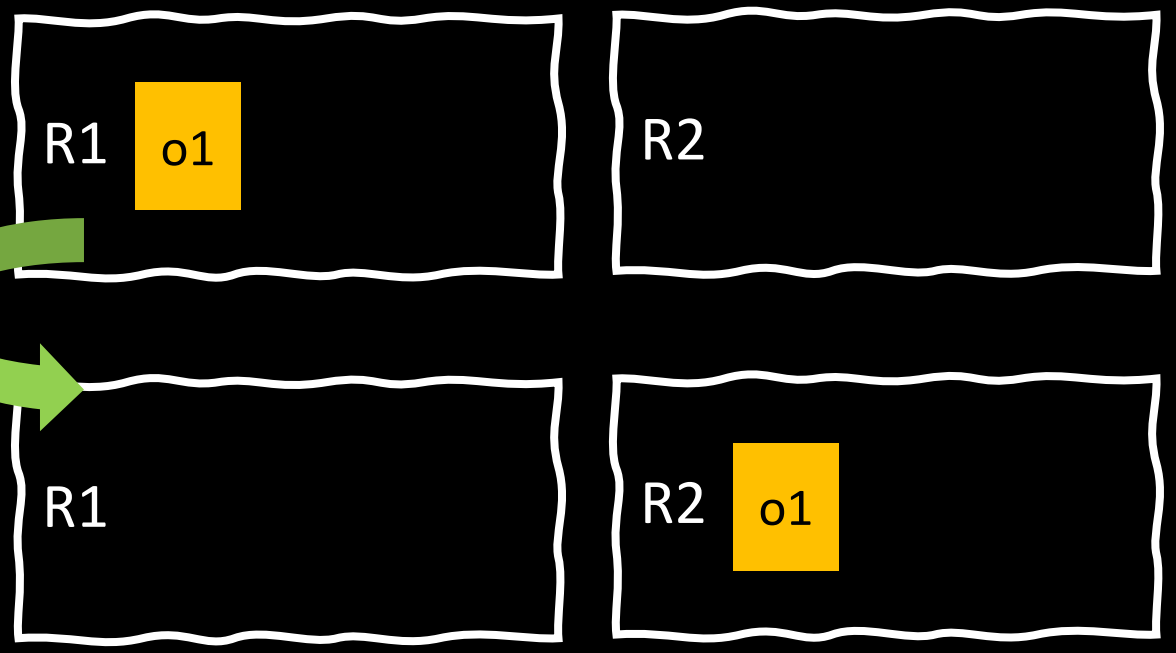


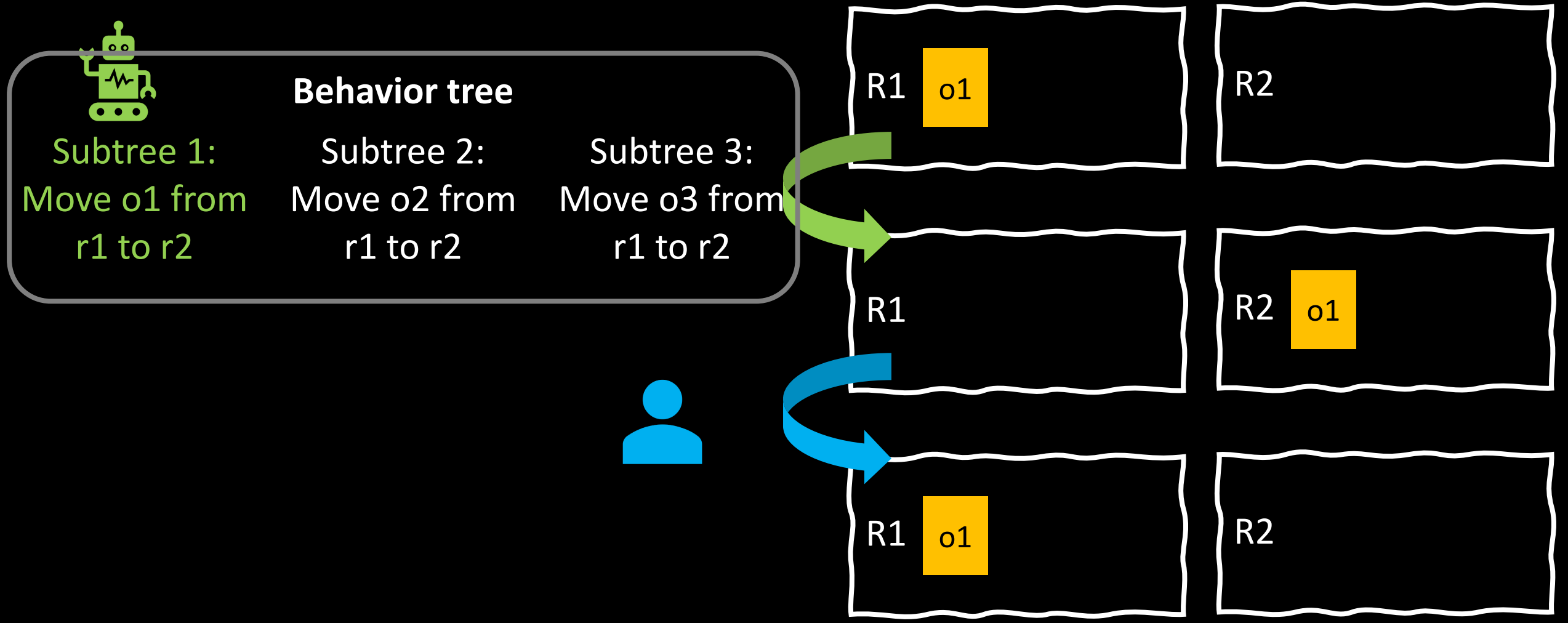
Behavior tree

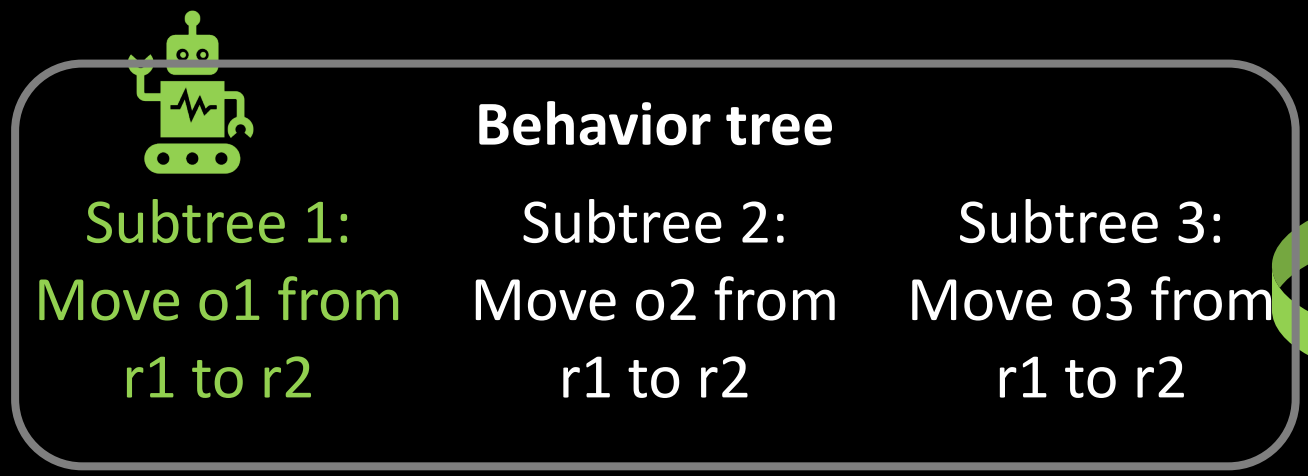
Subtree 1:
Move o1 from
r1 to r2

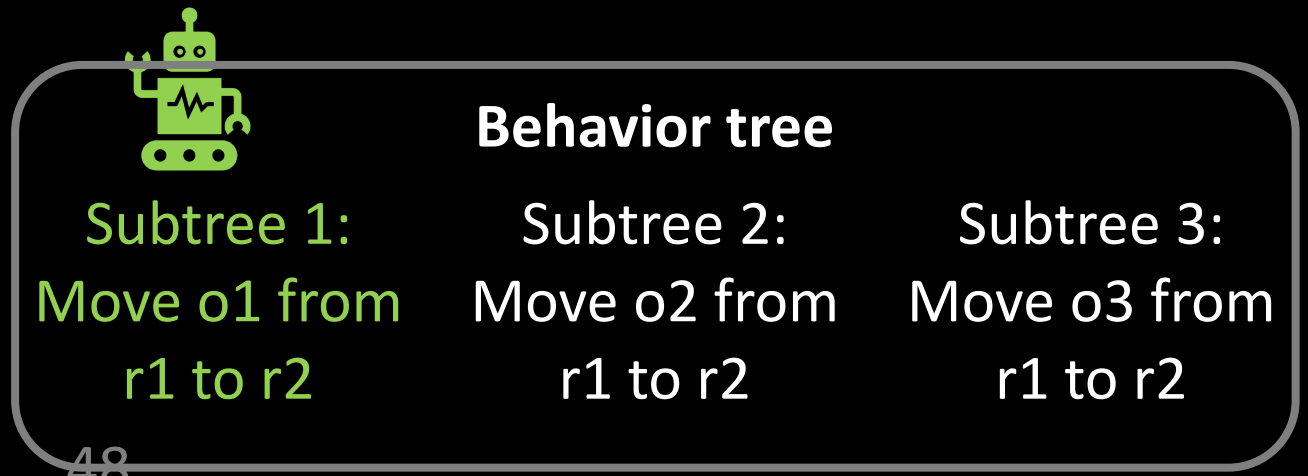
Subtree 2:
Move o2 from
r1 to r2

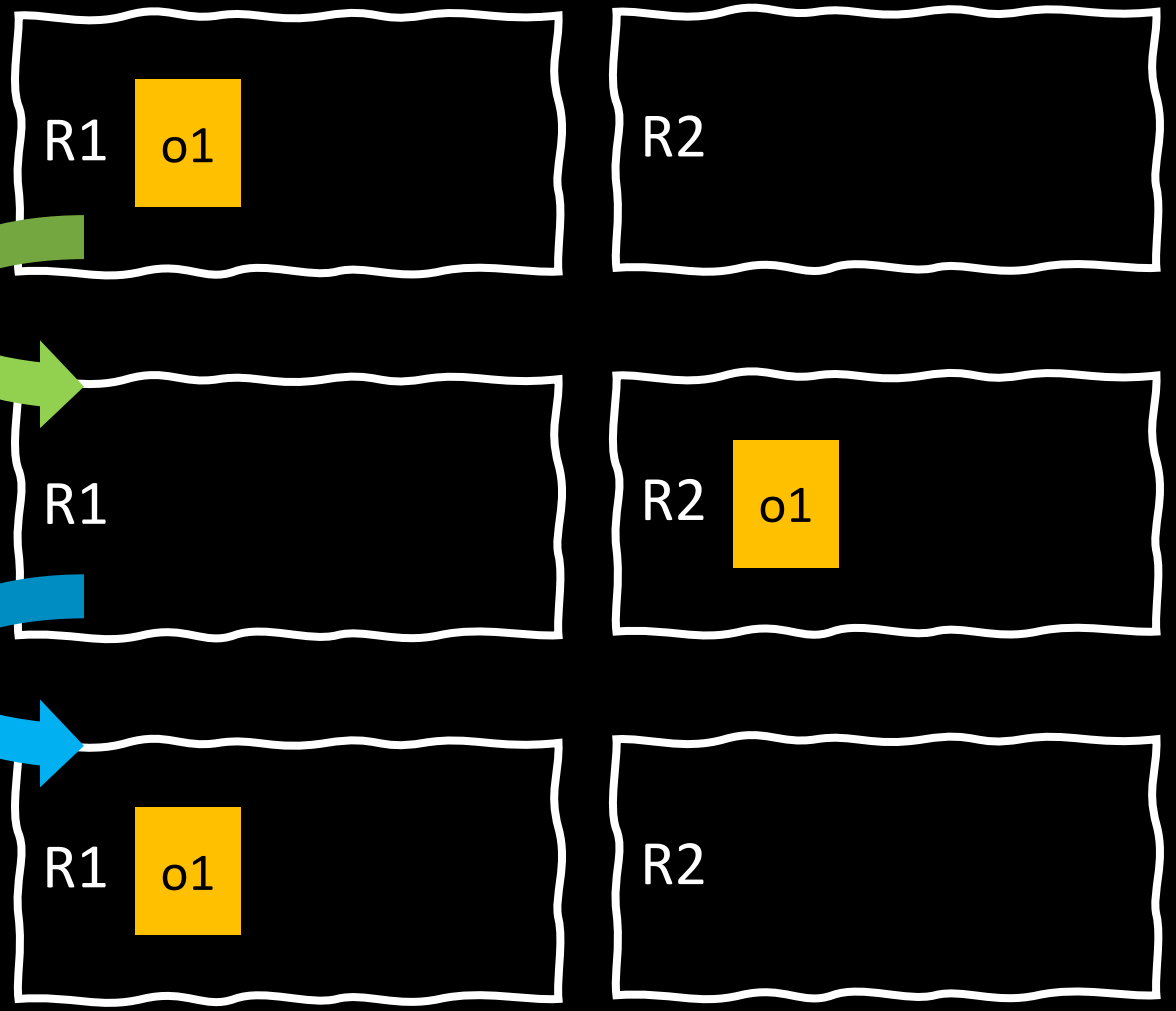
Subtree 3:
Move o3 from
r1 to r2

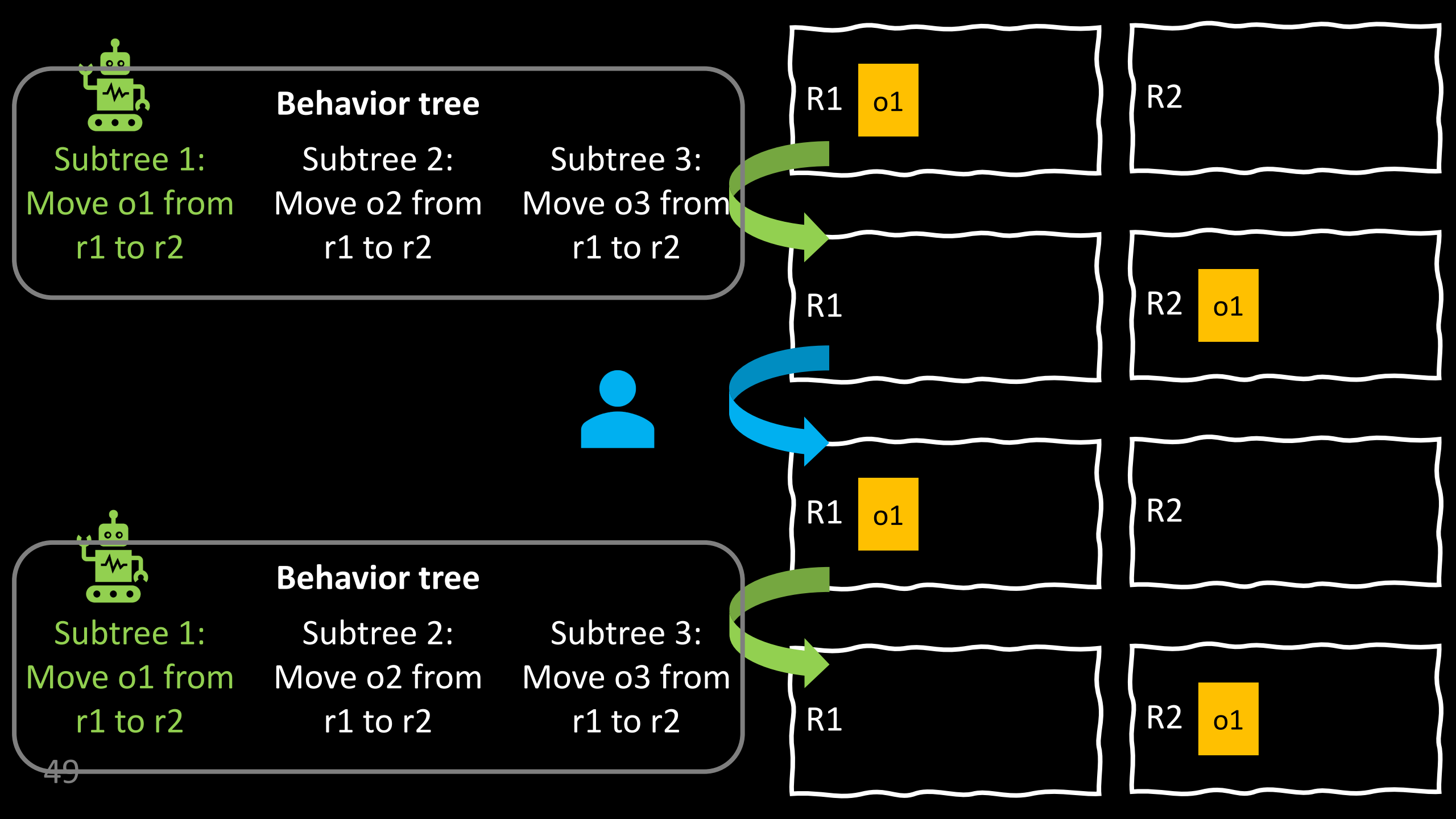


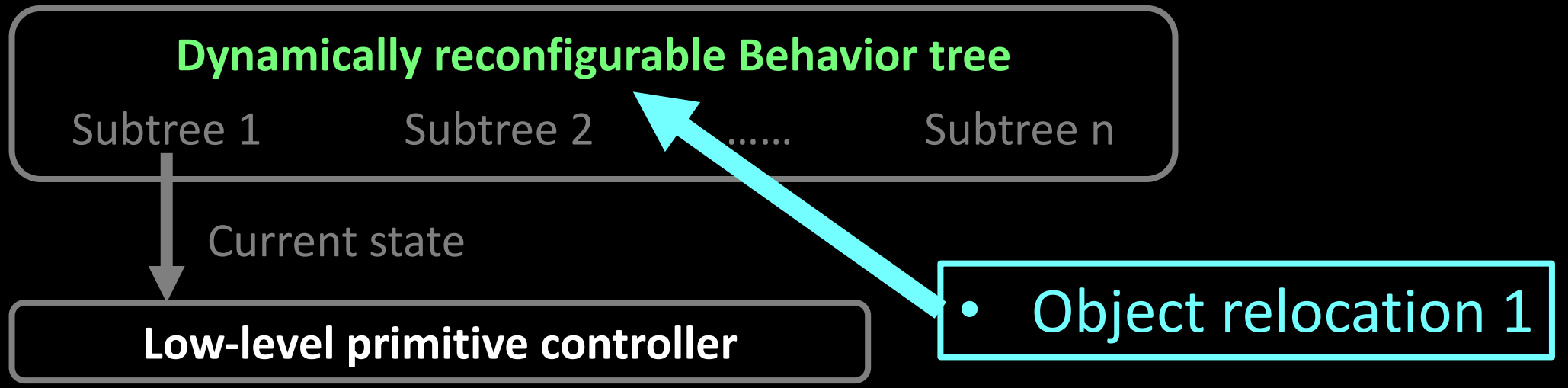

Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2
R1
o1
R2
R1
R2
o1
R1
o1
R2

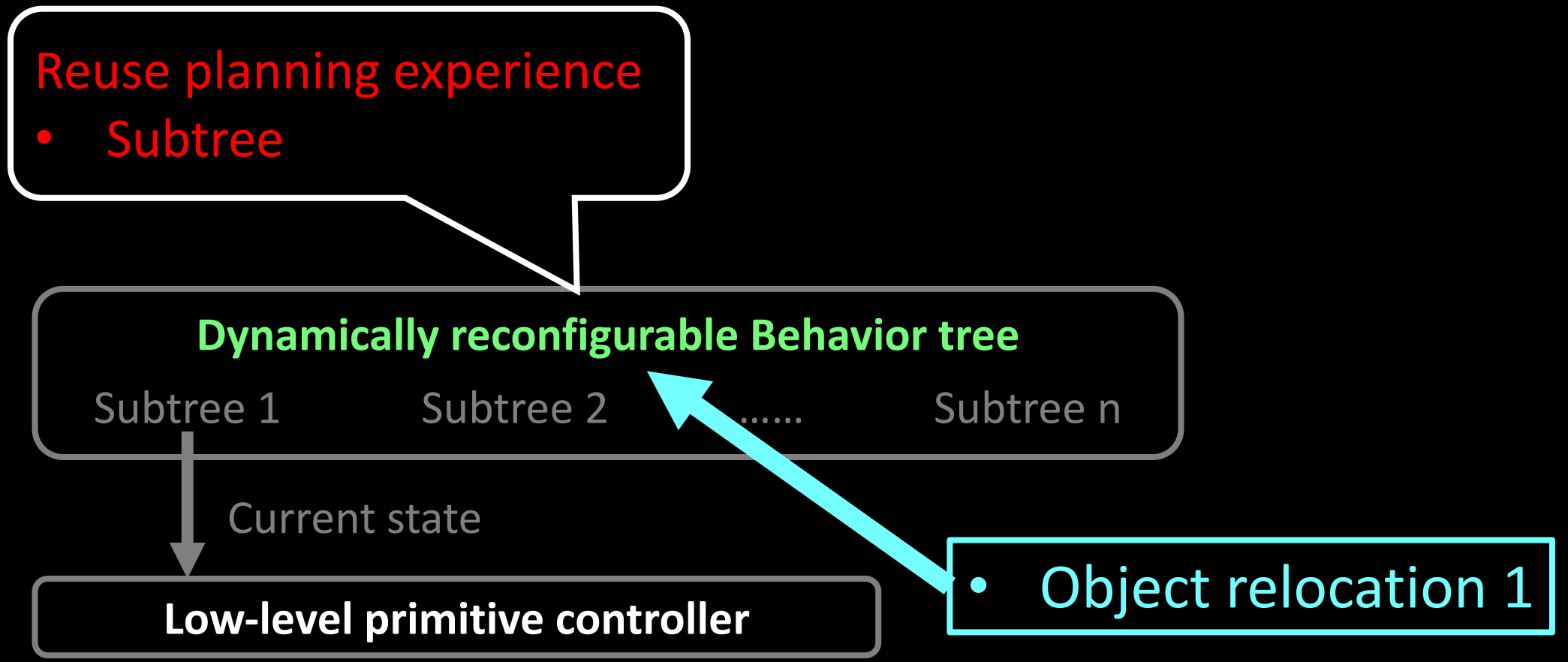

Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2

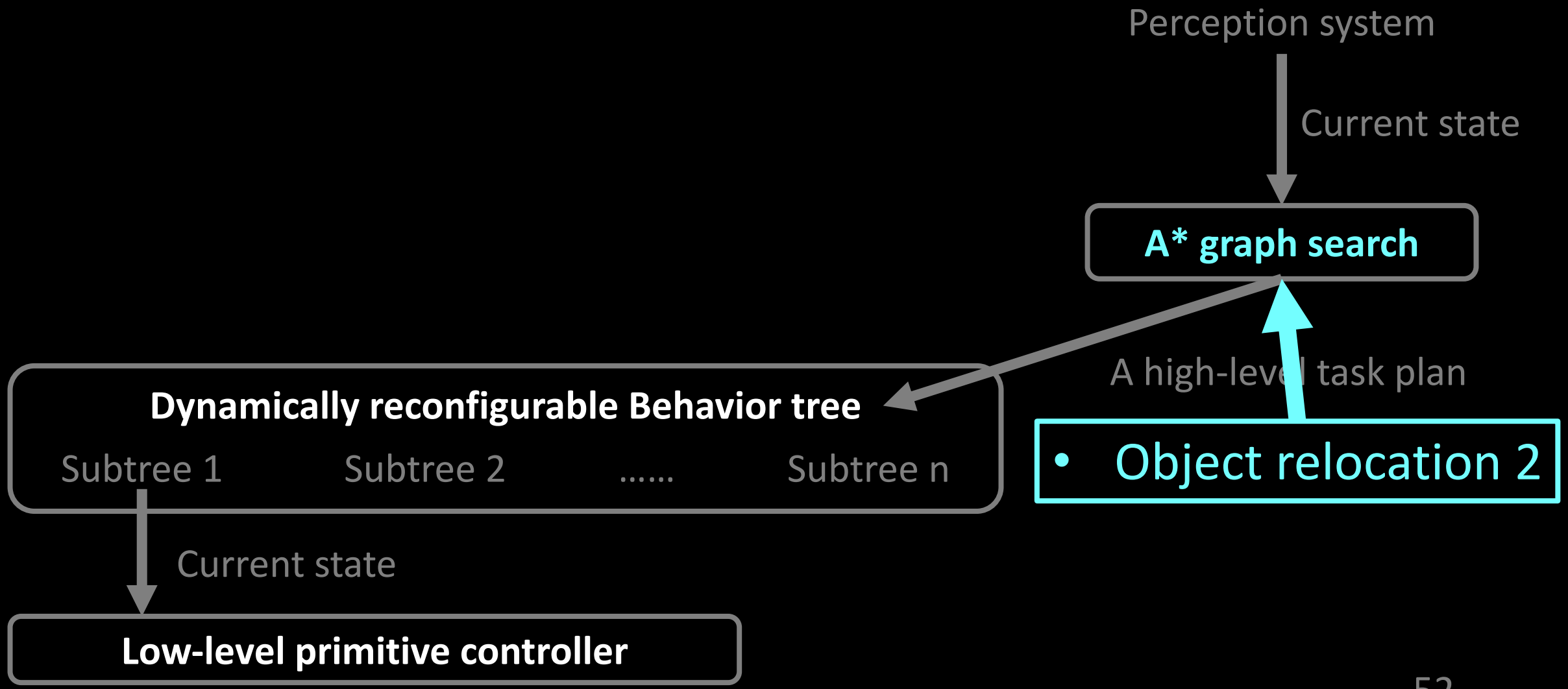

Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2


R1
o1
R2
R1
R2
o1
R1
o1
R2


Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2
R1
o1
R2
R1
R2
o1
R1
o1
R2
Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2
R1
R2
o1


Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller
• Object relocation 1


Reuse planning experience
• Subtree
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller
• Object relocation 1

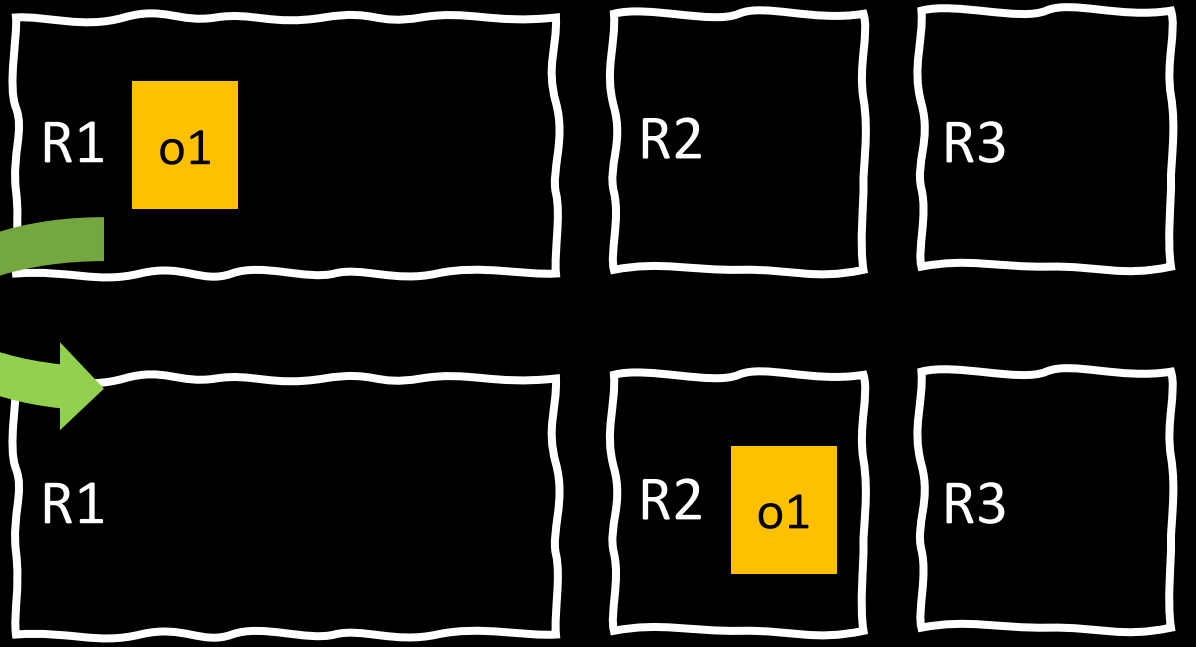

Perception system
Current state
A* graph search
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
• Object relocation 2
Current state
Low-level primitive controller

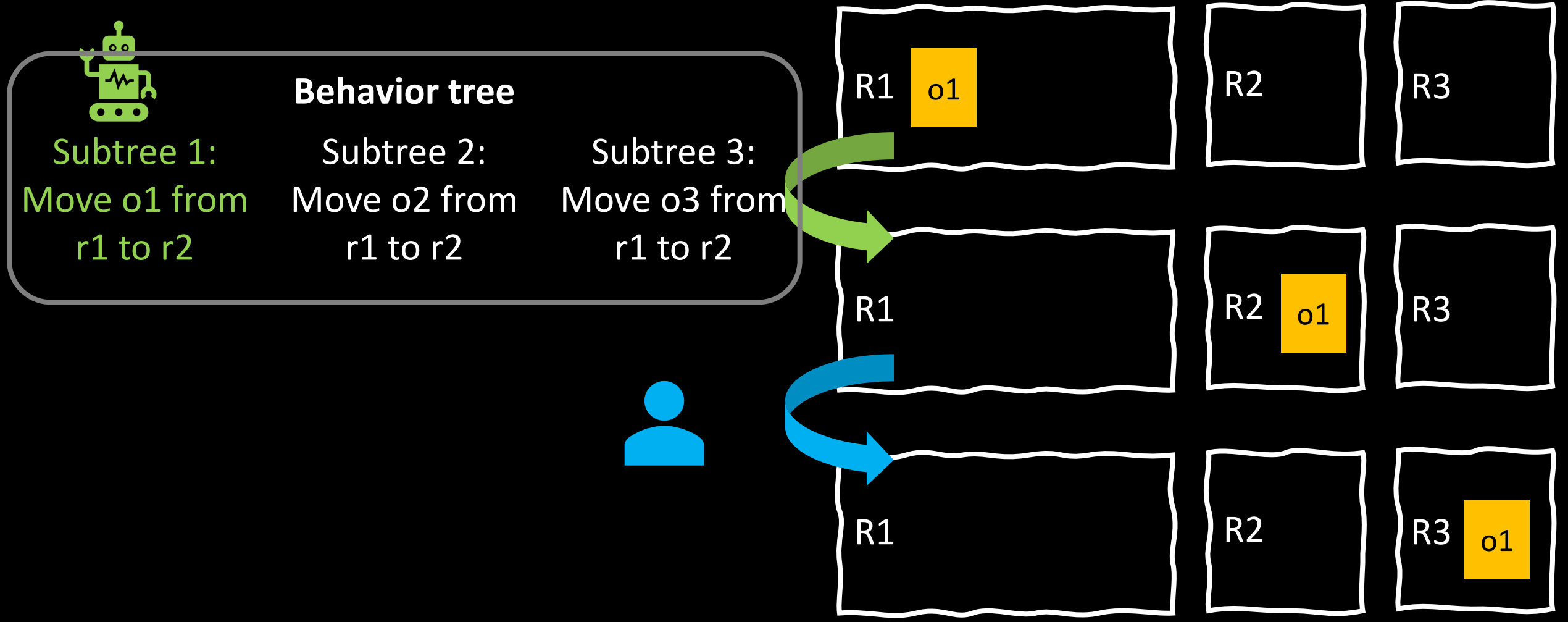
Behavior tree

Subtree 1: Move o1 from r1 to r2

Subtree 2: Move o2 from r1 to r2

Subtree 3: Move o3 from r1 to r2



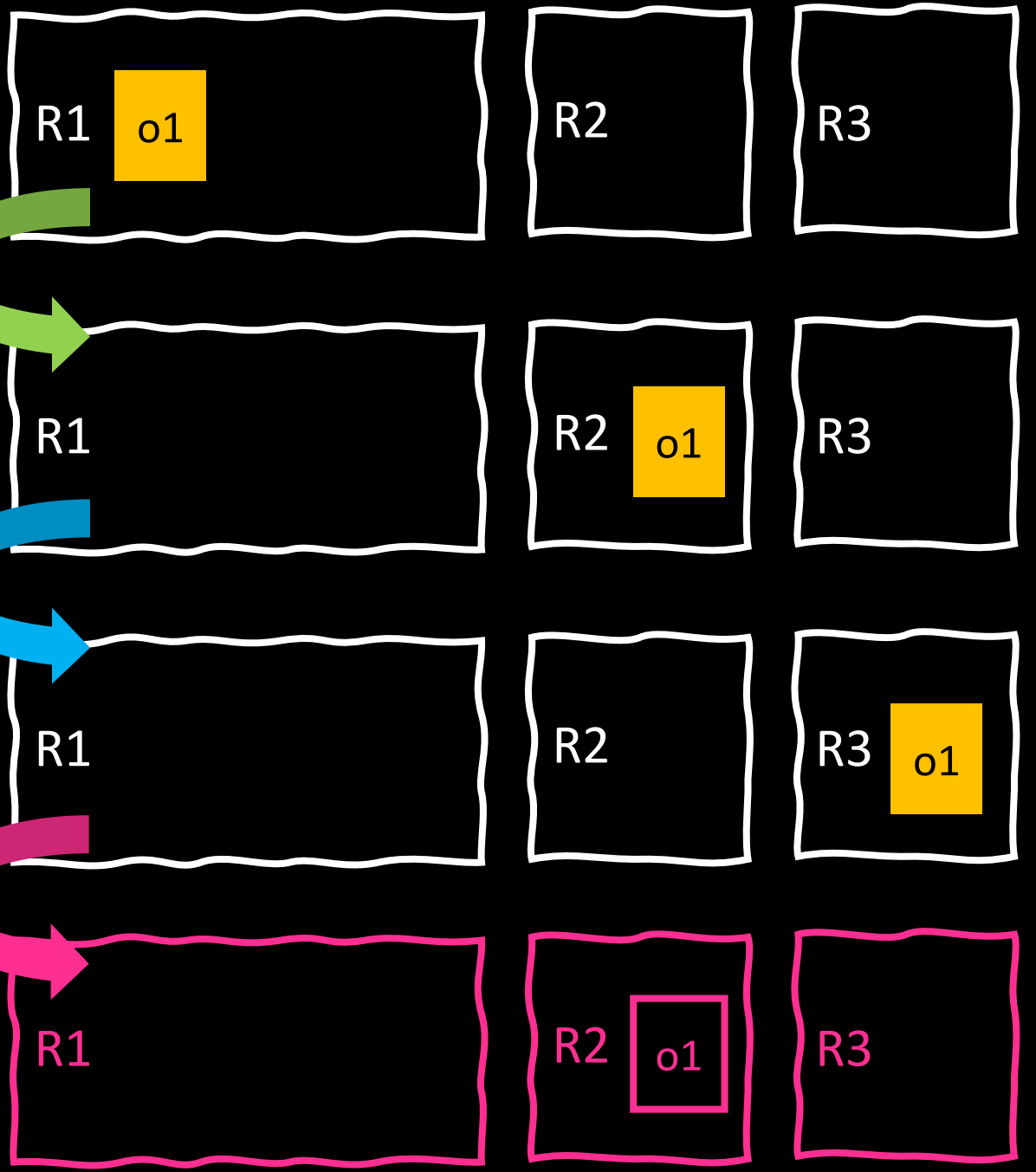

Behavior tree
Subtree 1:
Move o1 from
r1 to r2
Subtree 2:
Move o2 from
r1 to r2
Subtree 3:
Move o3 from
r1 to r2
R1
o1
R2
R3
R1
R2
o1
R3
R1
R2
R3
o1

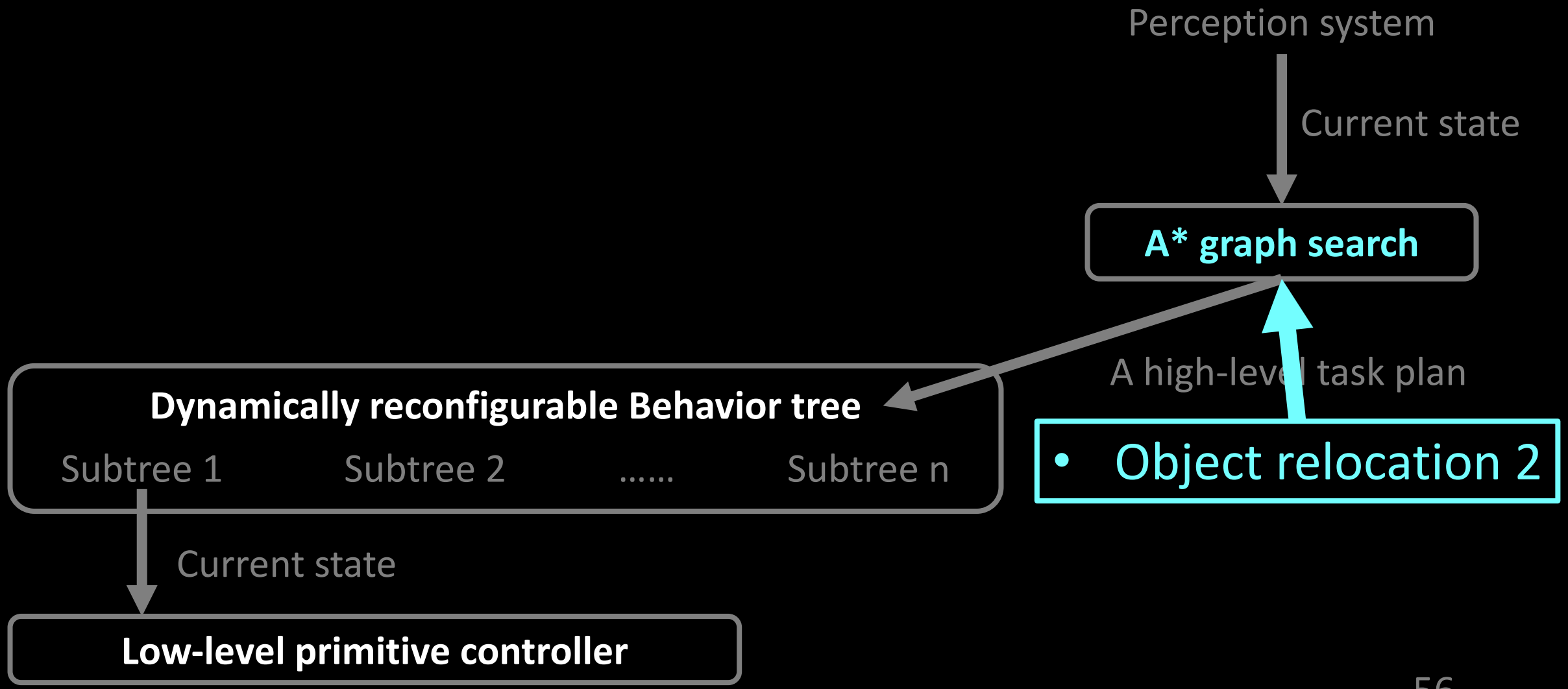
Behavior tree

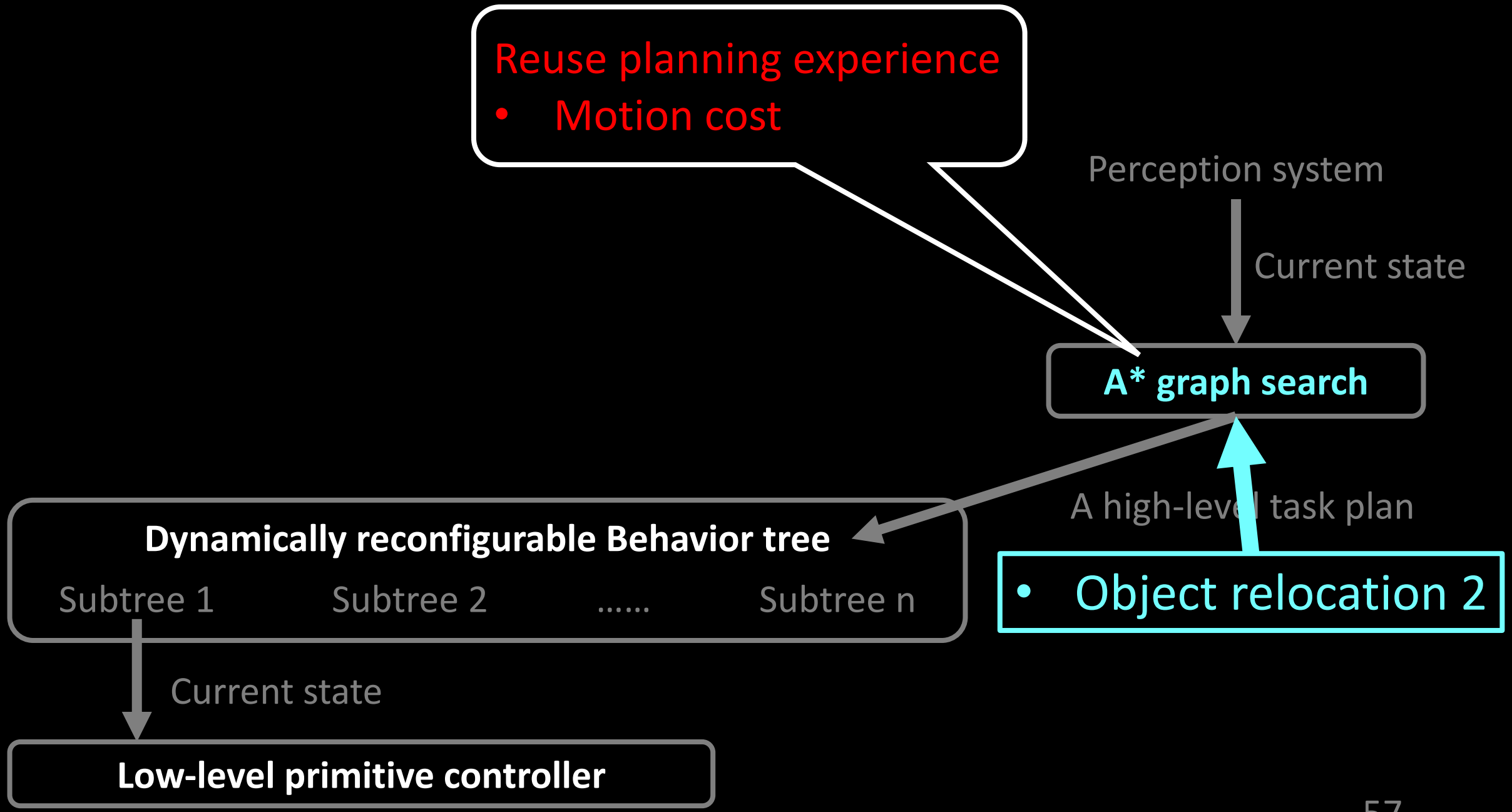
Subtree 1:
Move o1 from r1 to r2

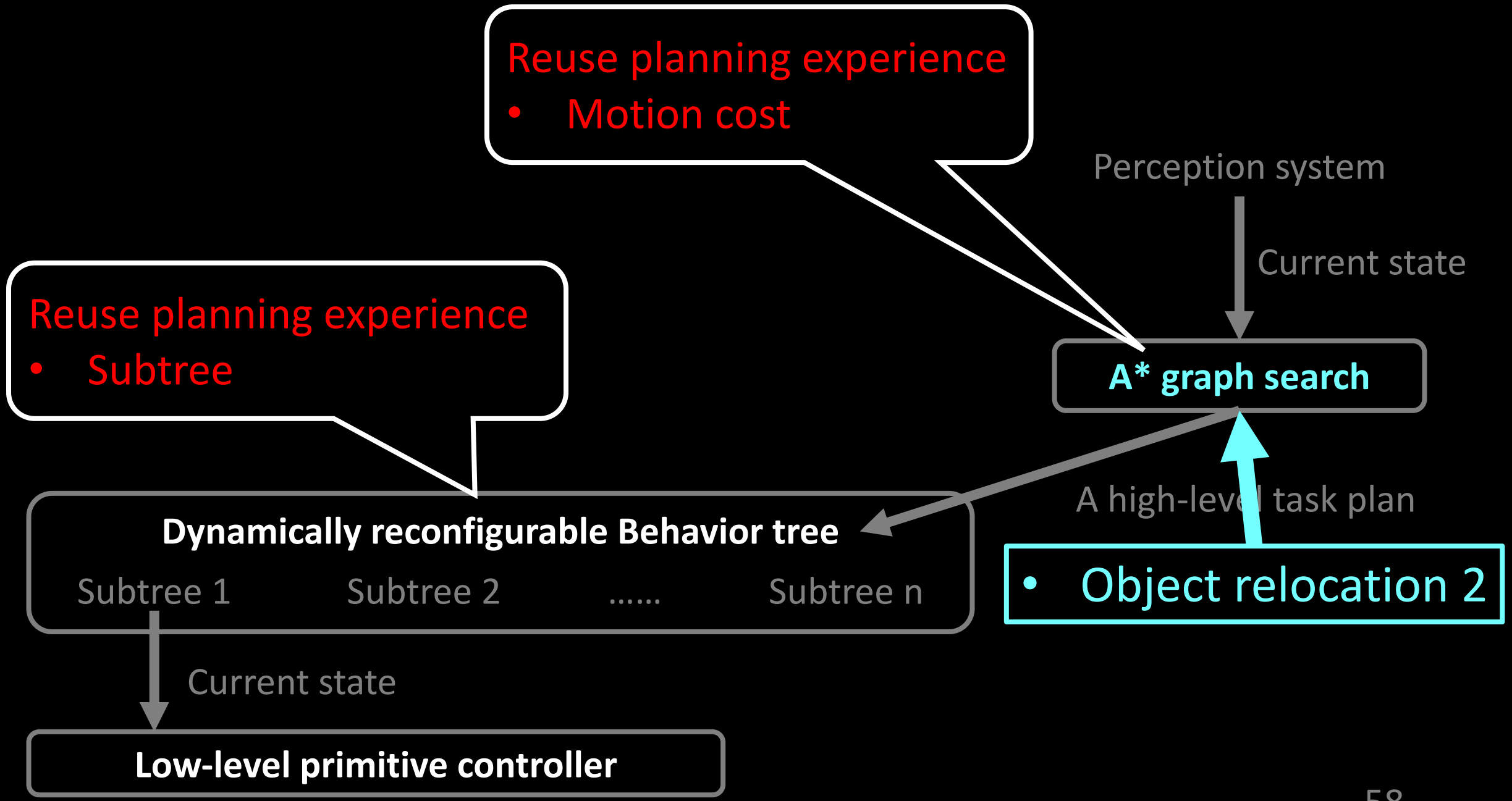
Subtree 2:
Move o2 from r1 to r2

Subtree 3:
Move o3 from r1 to r2

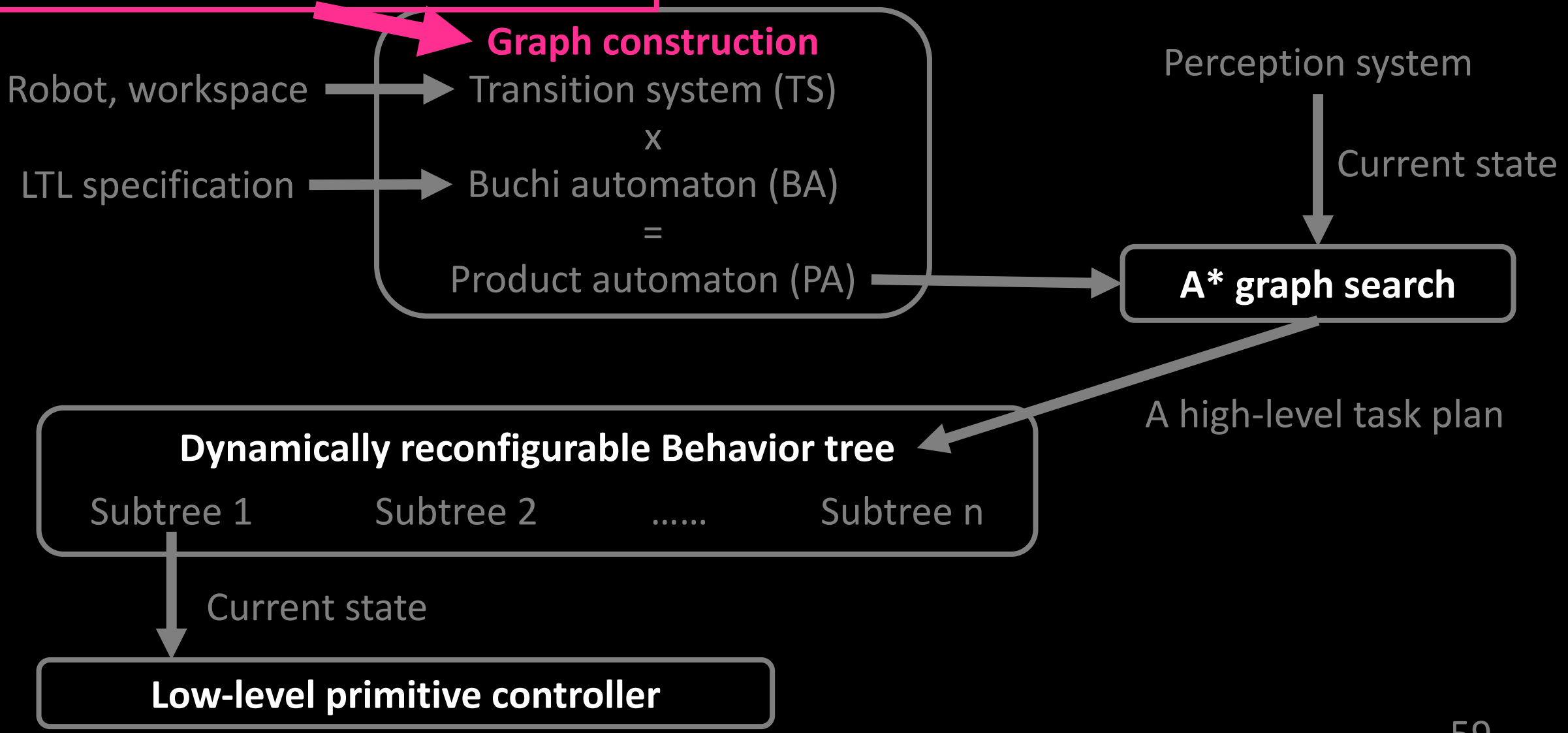


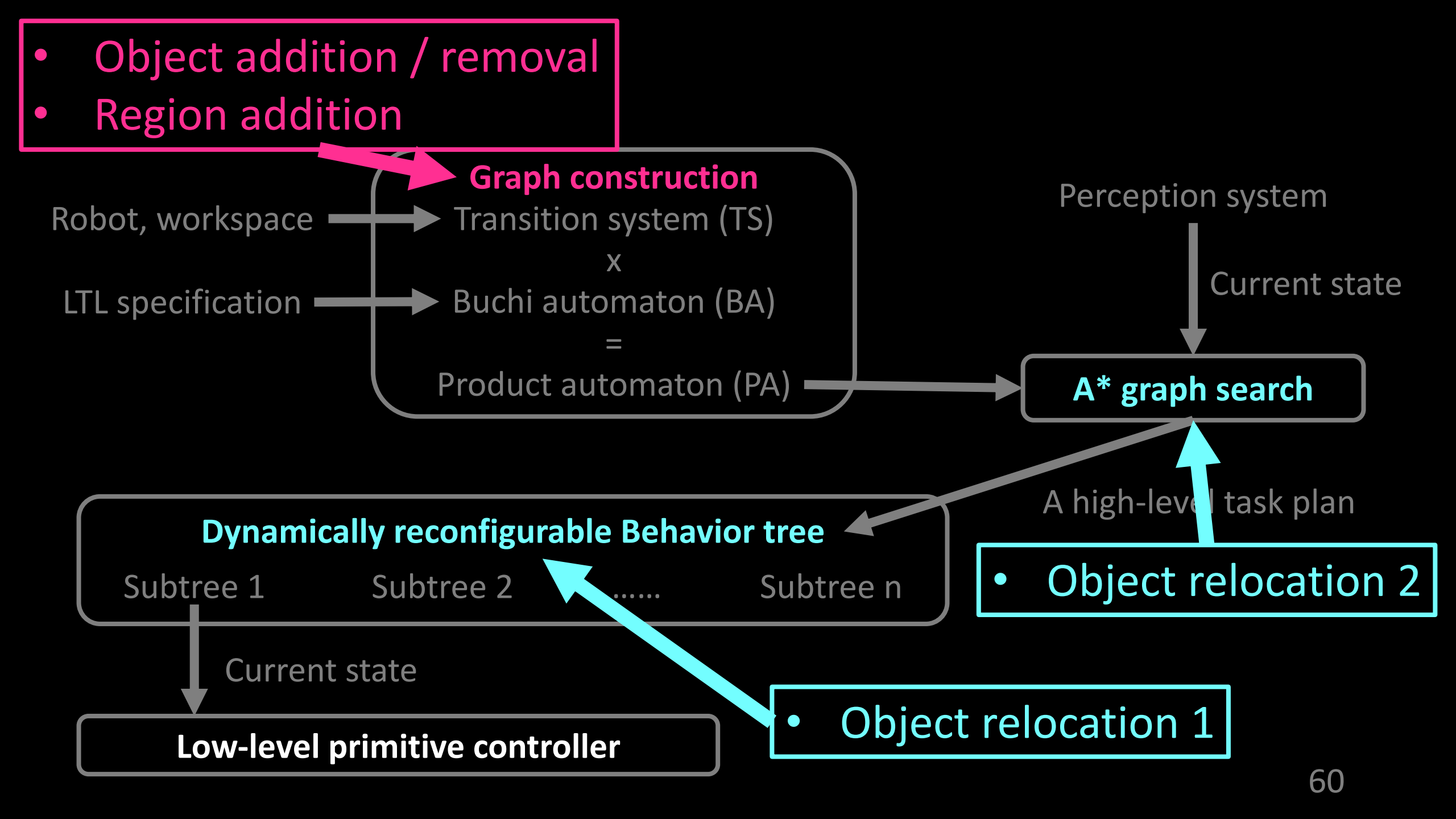

Perception system
Current state
A* graph search
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
• Object relocation 2
Current state
Low-level primitive controller


Reuse planning experience
• Motion cost
Perception system
Current state
A* graph search
A high-level task plan
• Object relocation 2
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
Current state
Low-level primitive controller


Reuse planning experience
• Motion cost
Perception system
Current state
Reuse planning experience
• Subtree
A* graph search
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
A high-level task plan
• Object relocation 2
Current state
Low-level primitive controller

- Object addition / removal
- Region addition



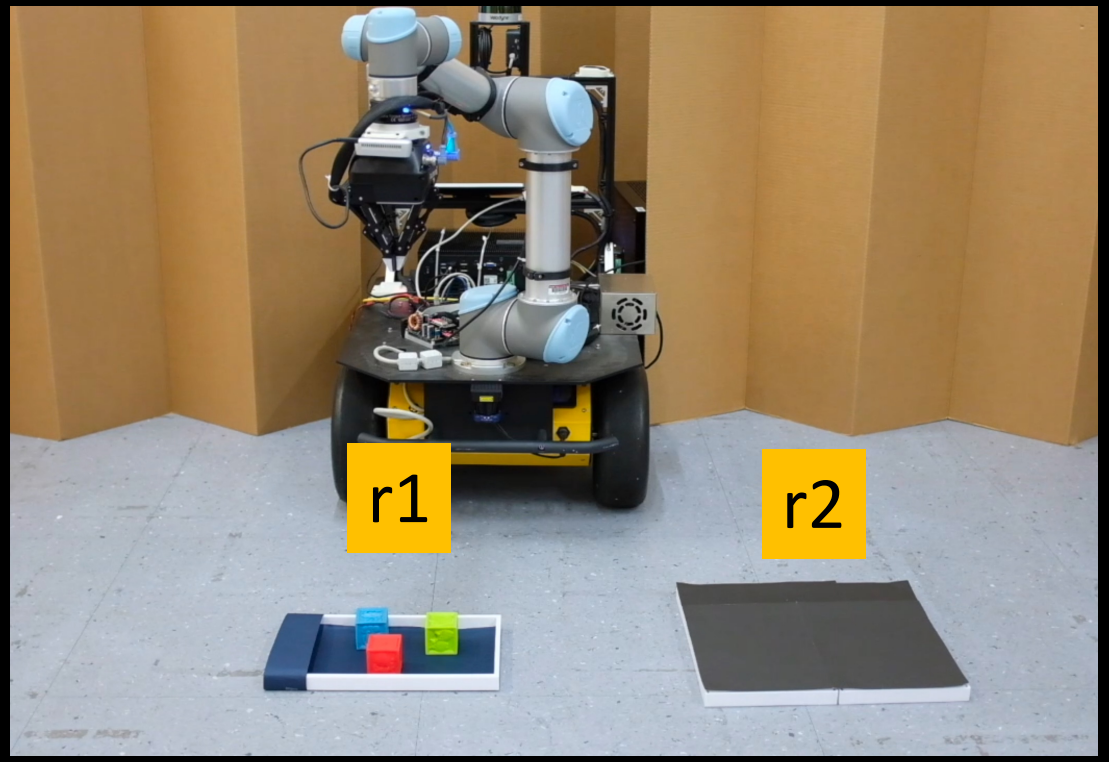

• Object addition / removal
• Region addition
Graph construction
Robot, workspace
LTL specification
Transition system (TS)
x
Buchi automaton (BA)
=
Product automaton (PA)
Perception system
Current state
A* graph search
A high-level task plan
Dynamically reconfigurable Behavior tree
Subtree 1
Subtree 2
……
Subtree n
• Object relocation 2
Current state
Low-level primitive controller
• Object relocation 1

Experiments

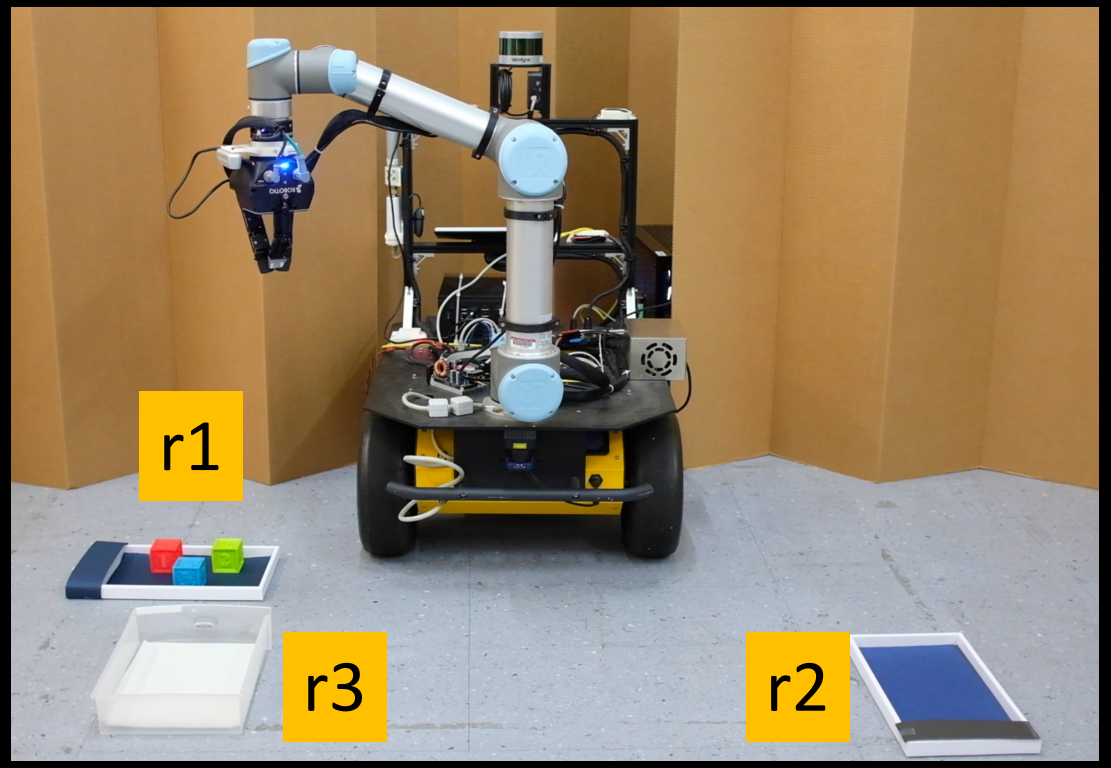
$$\mathcal{FG}(all\_obj\_in\_r_2)$$

- Random changes to env
  - Repositioning 1 object
  - Removing 1 object
  - Adding 1 object
  - Adding 1 region

Experiments $\mathcal{FG}(all\_obj\_in\_r_2)$



3 block



3 block + tray

Types of changes to the environment

	3-block + tray	Success	Init plan time (s)	Total replan time (s)	# replan	Comple time (s)
Object relocation	A*, exp	30/30	68.39 ± 1.00	**0.69 ± 0.43**	0.80, 0	**466.50 ± 34.80**
	A*	30/30	67.19 ± 0.04	33.50 ± 15.26	0.63, 0	530.17 ± 19.04
	Dijkstra	30/30	67.36 ± 0.04	30.40 ± 13.06	0.47, 0	561.57 ± 27.29
Object removal	A*, exp	30/30	67.62 ± 0.22	**0.10 ± 0.02**	1.93, 1	**394.48 ± 13.14**
	A*	30/30	67.34 ± 0.07	52.57 ± 8.18	1.30, 1	536.25 ± 31.49
	Dijkstra	30/30	67.57 ± 0.08	60.53 ± 9.16	1.40, 1	478.43 ± 17.54
Object addition	A*, exp	30/30	67.78 ± 0.32	**20.61 ± 0.86**	1.37, 1	**647.00 ± 24.67**
	A*	30/30	69.16 ± 1.29	89.60 ± 8.73	1.13, 1	734.74 ± 33.32
	Dijkstra	30/30	67.60 ± 0.10	116.16 ± 16.35	1.33, 1	720.81 ± 25.03

Proposed algorithm: A* + reuse experience

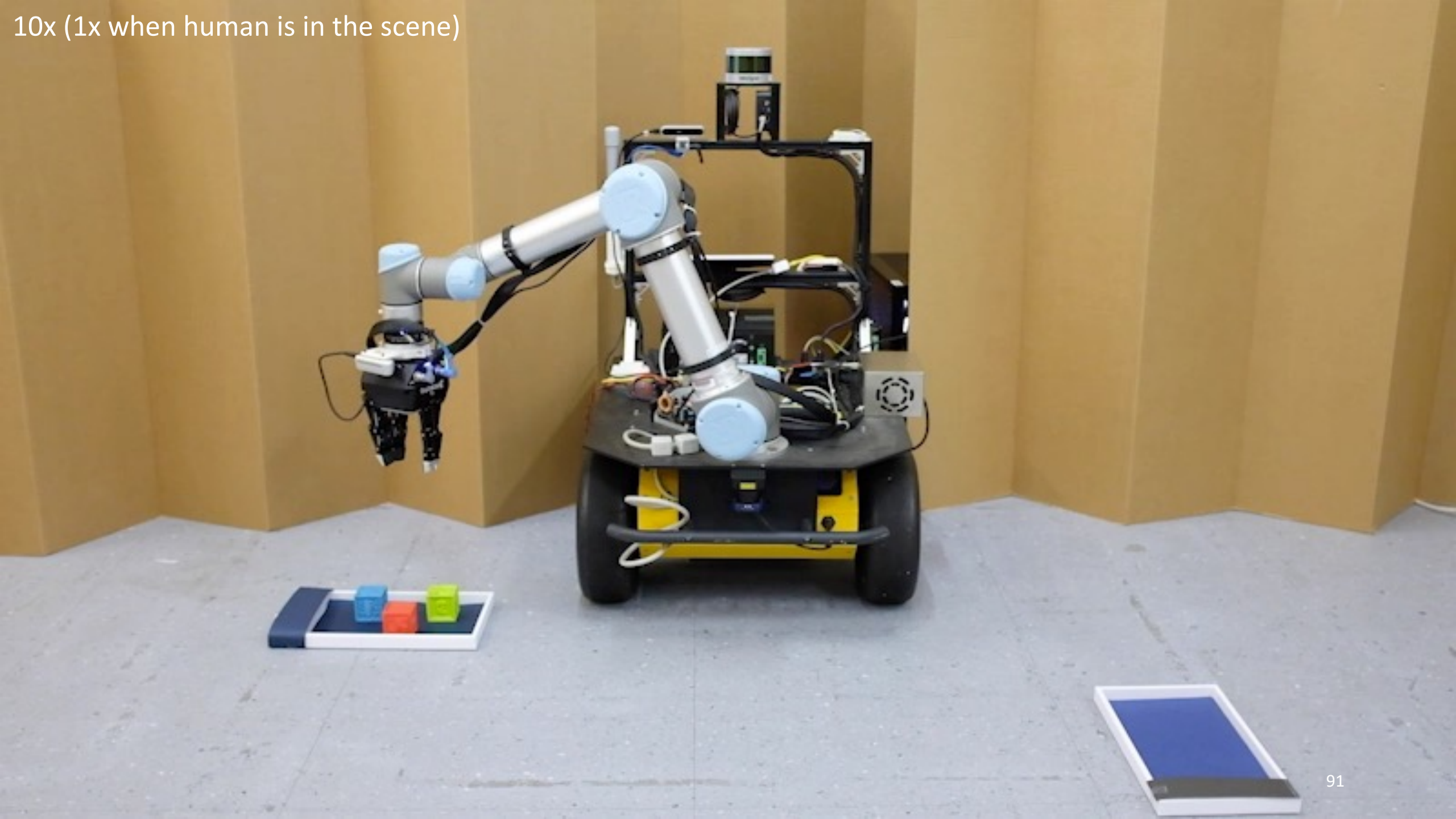
3-block + tray		Success	Init plan time (s)	Total replan time (s)	# replan	Comple time (s)
Object relocation	A*, exp	30/30	68.39 ± 1.00	**0.69 ± 0.43**	0.80, 0	**466.50 ± 34.80**
	A*	30/30	67.19 ± 0.04	33.50 ± 15.26	0.63, 0	530.17 ± 19.04
	Dijkstra	30/30	67.36 ± 0.04	30.40 ± 13.06	0.47, 0	561.57 ± 27.29
Object removal	A*, exp	30/30	67.62 ± 0.22	**0.10 ± 0.02**	1.93, 1	**394.48 ± 13.14**
	A*	30/30	67.34 ± 0.07	52.57 ± 8.18	1.30, 1	536.25 ± 31.49
	Dijkstra	30/30	67.57 ± 0.08	60.53 ± 9.16	1.40, 1	478.43 ± 17.54
Object addition	A*, exp	30/30	67.78 ± 0.32	**20.61 ± 0.86**	1.37, 1	**647.00 ± 24.67**
	A*	30/30	69.16 ± 1.29	89.60 ± 8.73	1.13, 1	734.74 ± 33.32
	Dijkstra	30/30	67.60 ± 0.10	116.16 ± 16.35	1.33, 1	720.81 ± 25.03

Hierarchy ~ increased efficiency

Number of calls to A* graph search

3-block + tray		Success	Init plan time (s)	Total replan time (s)	# replan	Comple time (s)
Object relocation	A*, exp	30/30	68.39 ± 1.00	**0.69 ± 0.43**	0.80, 0	**466.50 ± 34.80**
	A*	30/30	67.19 ± 0.04	33.50 ± 15.26	0.63, 0	530.17 ± 19.04
	Dijkstra	30/30	67.36 ± 0.04	30.40 ± 13.06	0.47, 0	561.57 ± 27.29
Object removal	A*, exp	30/30	67.62 ± 0.22	**0.10 ± 0.02**	1.93, 1	**394.48 ± 13.14**
	A*	30/30	67.34 ± 0.07	52.57 ± 8.18	1.30, 1	536.25 ± 31.49
	Dijkstra	30/30	67.57 ± 0.08	60.53 ± 9.16	1.40, 1	478.43 ± 17.54
Object addition	A*, exp	30/30	67.78 ± 0.32	**20.61 ± 0.86**	1.37, 1	**647.00 ± 24.67**
	A*	30/30	69.16 ± 1.29	89.60 ± 8.73	1.13, 1	734.74 ± 33.32
	Dijkstra	30/30	67.60 ± 0.10	116.16 ± 16.35	1.33, 1	720.81 ± 25.03

10x (1x when human is in the scene)



Reactive Task and Motion Planning under Temporal Logic Specifications

- Task and motion planning under environmental changes
  - Hierarchical system design for reactive behavior generation
- Efficiently handle environmental changes
  1. Hierarchical system design
  2. Algorithmic design

Reactive Task and Motion Planning under Temporal Logic Specifications

Shen Li*, Daehyung Park*, Yoonchang Sung*,

Julie A. Shah, Nicholas Roy

Massachusetts Institute of Technology

* These authors contributed equally to this work.

Robust Robotics Group