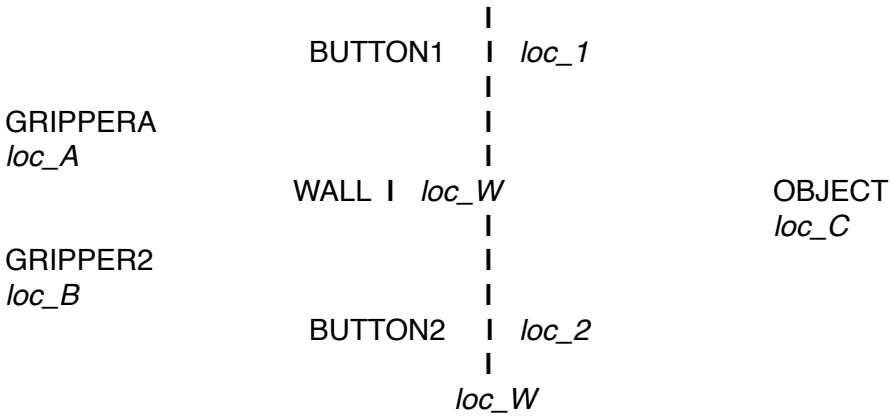
Spring Project - Baxter Robot

Simulation:



Description

- After each action a_i, an analysis s_i is performed to evaluate the current scenario. Immediately following that, a path trajectory p_i is generated

---

Observation step

Robot starts by making the following observation:

- SCENARIO [s_0]: *at*(gripperA, loc_A), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *at*(object, loc_C)
- GOAL [s_{goal}]: *at*(object, loc_A)
- [p_0]: *move_to*(gripperA, loc_C), *grasp*(gripperA, object), *move_to*(gripperA, loc_A)
- Available actions: [*push_button*(gripperX, buttonY) ∨ *move_to*(gripperX, loc_Z), *grasp*(gripperX, object)] with the following characterizations:

action	description	pressure states
push_button(gripperX, buttonY)	*find_button(buttonY)* *move_gripper(gripperX, loc_buttonY)* *apply_pressure_until_no_movevent(gripperX, buttonY)* *move_gripper(gripperX, loc_orig)*	*¬pressure(button)* *¬pressure(button)* *pressure(button, direction)* *¬pressure(button)*
move_to(gripperX, loc_Z)	*find_destination(loc_Z)* *move_to(gripperX, loc_Z)* *(circular definition?)*	*¬pressure(button)*

grasp(gripperX, object)	*find_object(object)* *move_gripper(gripperX, loc_object)* *close_gripper(gripperX)* *lift_arm()* *move_arm(above_loc_object)* *lower_arm_until_no_movement()* *open(gripperX)*	*¬pressure(object)* *¬pressure(object)* *¬pressure(object)* *¬pressure(object)* *¬pressure(object)* *¬pressure(object)* *pressure(object)*

Action and Observation step

[a_1]: *move_to*(gripperA, loc_C)

- FAILURE: at the point of failure, observe *at*(gripperA, loc_W), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *at*(object, loc_C)
- define this as an *obstruction* : *is_a*(wall, obstruction)

[s_1]: yields the following scenario observation, *at*(gripperA, loc_W), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)

[p_1] new path plan: ∅ since the preconditions of *move_to*(gripperA, loc_C) are not met due to the obstruction.

Reasoning step

- Now, chose from AVAILABLE ACTIONS: *push_button*(gripperX, buttonY), *move_to*(gripperX, loc_Z)
- Since *move_to* recently FAILED, try other action on both buttons and observe
- Note that *pre_*a_2 == s_1
- [a_2]: [*push_button*(gripperA, button1) ∨ *push_button*(gripperA, button2) ∨ *push_button*(gripperB, button1) ∨ *push_button*(gripperB, button2)]

Action and Observation step [experimental]

- [post_a_2 (gripperA, button_1) → s_2]: *at*(gripperA, loc_1), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [post_a_2 (gripperA, button_1) → p_2]: ∅ since the preconditions of *move_to*(gripperA, loc_C) are not met due to the obstruction.
- [post_a_2 (gripperA, button_2) → s_2]: *at*(gripperA, loc_2), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [post_a_2 (gripperA, button_2) → p_2]: ∅ since the preconditions of *move_to*(gripperA, loc_C) are not met due to the obstruction.
- [post_a_2 (gripperB, button_1) → s_2]: *at*(gripperB, loc_1), *at*(gripperA, loc_W), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [post_a_2 (gripperA, button_1) → p_2]: ∅ since the preconditions of *move_to*(gripperB, loc_C) are not met due to the obstruction.
- [post_a_2 (gripperA, button_2) → s_2]: *at*(gripperB, loc_2), *at*(gripperA, loc_W), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [post_a_2 (gripperA, button_2) → p_2]: ∅ since the preconditions of *move_to*(gripperB, loc_C) are not met due to the obstruction.

Reasoning step

- So nothing changes with each. In each case, we still have $s_2 \rightarrow p_2 = \varnothing$]::
- Perform *during* observations. We want to break up this primitive further

---

Action and Observation step [experimental]

- [during_a_2 (gripperA, button_1) → s_2]: *at*(gripperA, loc_1), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [during_a_2 (gripperA, button_2) → s_2`]: *at*(gripperA, loc_2), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *at*(object, loc_C)
- [during_a_2 (gripperB, button_1) → s_2]: *at*(gripperB, loc_1), *at*(gripperA, loc_W), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
- [during_a_2 (gripperB, button_2) → s_2`]: *at*(gripperB, loc_2), *at*(gripperA, loc_W), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *at*(object, loc_C)

Reasoning step

So at this point, we have collected the following data about a_2 :

- experimentally chose (gripperA, button_1) → pre_a_2 , during_a_2 , post_a_2 , $s_2 = \varnothing$
- experimentally chose (gripperA, button_2) → pre_a_2 , during_a_2 , post_a_2 , $s_2 = \varnothing$
- experimentally chose (gripperB, button_1) → pre_a_2 , during_a_2 , post_a_2 , $s_2 = \varnothing$
- experimentally chose (gripperB, button_2) → pre_a_2 , during_a_2 , post_a_2 , $s_2 = \varnothing$

but our algorithm has the ability to segment primitives based on changes in scenario's. That is, for any action primitive a_i if (during_a_i → s_i) ∧ (post_a_i → s_i`) ∧ ($s_i$` ≠ s_i) holds, then we can segment a_i in the following way:

a_{i_a} : *pre_a*$_{i\_a}$ = (pre_a_i → s_{i-1}), *post_a*$_{i\_a}$ = (during_a_i → s_i`)
a_{i_b} : *pre_a*$_{i\_b}$ = (during_a_i → s_i), *post_a*$_{i\_b}$ = (post_a_i → s_i)

or

a_{i_a} : *pre_a*$_{i\_a}$ = s_{i-1}, *post_a*$_{i\_a}$ = s_i`
a_{i_b} : *pre_a*$_{i\_b}$ = s_i, *post_a*$_{i\_b}$ = s_i

So for our example, using this philosophy with our experiential action results, we have:

- ¬*is_segmentable*(a_2(gripperA, button_1))
- *is_segmentable*(a_2(gripperA, button_2))
- ¬*is_segmentable*(a_2(gripperB, button_1))
- *is_segmentable*(a_2(gripperB, button_2))

Next step is to partition a_2(gripperA, button_2) = *push_button*(gripperA, button2), in a way that generates the following choices for actions:

- [a_3]: [*push_button*(gripperA, button1) ∨ *push_button*(gripperA, button2) ∨ *push_button_firstPartition*(gripperA, button2) ∨ *push_button_secondPartition*(gripperA, button2)]

with the following properties

action	pre_cond	during_cond	post_cond = scenario	path planning
push_button(gripperA, button1)	*at(gripperA, loc_W)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	*at(gripperA, loc_1)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	*at(gripperA, loc_1)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	∅ since is_a(wall, obstruction)
push_button(gripperA, button2)	*at(gripperA, loc_W)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	*at(gripperA, loc_2)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *¬is_a(wall, obstruction)* *at(object, loc_C)*	*at(gripperA, loc_2)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	∅ since is_a(wall, obstruction)
push_button_firstPartition(gripperA, button2)	*at(gripperA, loc_W)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	*Did not test*	*at(gripperA, loc_2)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *¬is_a(wall, obstruction)* *at(object, loc_C)*	*move_to(gripperA, loc_C)*

push_button_secondPartition(gripperA, button2)	*at(gripperA, loc_2)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *¬is_a(wall, obstruction)* *at(object, loc_C)*	*Did not test*	*at(gripperA, loc_2)* *at(gripperB, loc_B)* *at(wall, loc_W)* *at(button1, loc_1)* *at(button2, loc_2)* *is_a(wall, obstruction)* *at(object, loc_C)*	∅ since is_a(wall, obstruction)

therefore, we select *push_button_firstPartition*(gripperA, button2) to generate p_2 = *move_to*(gripperA, loc_C)

If all went well the script would be:

- [a_1]: *move_to*(gripperA, loc_C)
- [a_2]: *push_button_firstPartition*(gripperA, button2)
- [a_3]: *move_to*(gripperA, loc_C)

* The same holds for gripperB

Action and Observation step

[a_2]: *push_button_firstPartition*(gripperA, button2)
[s_2]: *at*(gripperA, loc_2), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *at*(object, loc_C)
[p_2] new path plan: *move_to*(gripperA, loc_C)

Action and Observation step

[a_3]: *move_to*(gripperA, loc_C)

- FAILURE: at the point of failure, observe *at*(gripperA, loc_2), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C),
- something here is different than what we had in s_2

[s_3]: *at*(gripperA, loc_2), *at*(gripperB, loc_B), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), *is_a*(wall, obstruction), *at*(object, loc_C)
[p_3] new path plan: ∅

If we look at our first table, with pressure exertion, we see that at the end of *push_button_firstPartition*(gripperA, button2), we have *pressure*(button2, direction), but at the point of failure, at the beginning of execution of *move_to*(gripperA, loc_C), we have ¬*pressure*(button2). So we need to swap out this primitive with something that accomplishes this predicate.

From our repertoire of available actions, we can substitute *push_button_firstPartition*(gripperA, button2) with *push_button_firstPartition*(gripperB, button2) (same desired post condition)

Action and Observation step

[a_4]: *push_button_firstPartition*(gripperB, button2)
[s_4]: *at*(gripperA, loc_2), at(), at(gripperB, loc_2), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *pressure*(button_2), *at*(object, loc_C)
[p_4] new path plan: *move_to*(gripperA, loc_C)

Action and Observation step
[a_5]: *move_to*(gripperA, loc_C)
[s_5]: *at*(gripperA, loc_C), *at*(gripperB, loc_2), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *pressure*(button_2), *at*(object, loc_C)
[p_5] new path plan: *grasp*(gripperA, object)

Action and Observation step
[a_6]: *grasp*(gripperA, object)
[s_6]: *at*(gripperA, loc_C), *at*(gripperB, loc_2), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *pressure*(button_2), *at*(object, loc_C)
[p_6] new path plan: *move_to*(gripperA, loc_A)

Action and Observation step
[a_7]: *move_to*(gripperA, loc_A)
[s_7]: *at*(gripperA, loc_A), *at*(gripperB, loc_2), *at*(wall, loc_W), *at*(button1, loc_1), *at*(button2, loc_2), ¬*is_a*(wall, obstruction), *pressure*(button_2), *at*(object, loc_A)
[p_7] new path plan: ∅, GOAL REACHED

**NOTE that in this implementation, the object is still being grasped and the button is still being pressed*