Algorithm Behavior Tree Generator from a PDDL Plan - P3 Assembler - (Part 1)

```
1:  Input: PDDL plan file plan.pddl, Knowledge base, Output file path
2:  Output: Python file behavior_tree_autoV2.py (generated Behavior Tree)
3:  Initialize knowledge_base with operations and marker IDs
4:  Initialize ALL_PREDICATE_PARAMETER_ORDER with the parameter order for each predicate
5:  Initialize opposite_location to map opposite locations
6:  function parse_pddl_plan(plan_file)
7:      Initialize an empty dictionary actions
8:      Initialize an empty set tools_in_plan
9:      Initialize an empty dictionary tool_to_op
10:     Define a regex pattern to extract actions
11:     Open plan_file for reading
12:     for each line in the file do
13:         Clean line (remove extra spaces)
14:         if line is empty or starts with ';' then
15:             Continue to next iteration
16:         end if
17:         Apply pattern to extract time, action_name, params_str
18:         if pattern matches then
19:             Extract params by splitting params_str
20:             if params is empty then
21:                 Display a warning and Continue
22:             end if
23:             action_key = action_name + time
24:             if action_name contains "MOVE_TO" then
25:                 if number of parameters != 3 then
26:                     Display an error and Continue
27:                 end if
28:                 Extract agent, from_location, to_location from params
29:                 Add this action to actions
30:             else if action_name contains "PICK" or "PLACE" then
31:                 Extract agent, tool from params
32:                 Add tool to tools_in_plan
33:                 Extract op_key from action_name or tool
34:                 if op_key found then
35:                     tool_to_op[tool] = op_key
36:                 else
37:                     Display a warning and Continue
38:                 end if
39:                 if (action_name contains "PICK" and param count = 3) or (action_name contains "PLACE" and param count = 3) then
40:                     location = params[2]
41:                 else
42:                     location = None
43:                 end if
44:                 Add this action to actions
45:             else if action_name contains "WAIT" then
46:                 Extract agent from params
47:                 Add this action to actions
48:             else
49:                 Display a warning
50:             end if
51:         end if
52:     end for
53:     Return (actions, tool_to_op)
54: end function
55: function filter_actions(actions_dict)
56:     Initialize an empty dictionary filtered_actions
57:     for each key, action in actions_dict do
58:         if action type = "move_to" and from = to then
59:             Continue to next iteration
60:         end if
61:         Add action to filtered_actions
62:     end for
63:     Return filtered_actions
64: end function
65: function get_base_op_key(op_key)
66:     Return op_key.split('_')[0]
67: end function
```

Algorithm Behavior Tree Generator from a PDDL Plan - P.3 (Part 2)

```
function create_behavior_tree_file(actions_dict, tool_to_op, output_file)
    Collect unique operations from tool_to_op into unique_operations
    Initialize an empty set used_predicates
    Update used_predicates based on actions
    Open output_file for writing
    Write the BT file header
    Define PREDICATE_PARAMETER_ORDER with the used predicates
    Define a function update_kb for KB updates
    Define a class KBUpdateDecorator for decorators
    Begin create_behavior_tree()
    Initialize root sequence root
    Define marker_id_pick_OPXX, marker_id_place_OPXX
    Initialize an empty list decorators
    for each op_name, details in actions_dict do
        if action type = "move_to" then
            Generate move_sequence
            Generate move_decorator
            Add move_decorator to decorators
        else if action type = "wait" then
            Generate wait_sequence
            Generate wait_decorator
            Add wait_decorator to decorators
        else if action type = "pick" or "place" then
            Extract tool, op_key, etc.
            if action = "pick" then
                Generate pick_sequence
                Generate pick_decorator
            else if action = "place" then
                Generate place_sequence
                Generate place_decorator
            end if
            Add the corresponding decorator to decorators
        end if
    end for
    Add all decorators to the root sequence root
    Return root
end function
Main Program:
Define plan_file (path to PDDL plan)
(actions_dict, tool_to_op) = parse_pddl_plan(plan_file)
filtered_actions = filter_actions(actions_dict)
Define output_file
create_behavior_tree_file(actions_dict, tool_to_op, output_file)
Display "Behavior Tree file output_file generated successfully."
```