Algorithm Dispatch Elementary Actions to the PDDL Knowledge Base - P2 Dispatcher - (Part 1)

```
Input:
    1. JSON file (operations_elementaires.json)
    2. Log file (solution_readable.txt)
Output: domain.pddl, problem.pddl
Load operations_elementaires.json into operations_elementaires
Load solution_readable.txt into logs
Initialize:
    operations = empty
    operations_requiring_tools = empty
    move_to_operations = empty
    tools = empty
    filtered_logs = empty
    locations = empty
    op_locations = empty
    location_counter = 1
    previous_end_time = None
    previous_op_name = None
for each log in logs do
    Extract (op_id, job_id, resource, start_time, end_time)
    if op_id and job_id contain '_' then
        op_num = get_number(op_id)
        job_num = get_number(job_id)
        suffix = ""
        if resource = "Co" then
            suffix = "_CO"
        end if
        op_name = "OP" + job_num + op_num + suffix
        Add (op_name, job_id, resource, start_time, end_time) to operations
        Add (op_name, job_id, resource, start_time, end_time) to filtered_logs
        if previous_end_time not None and start_time ¿ previous_end_time then
            Add ("wait", job_id, resource, previous_end_time, start_time) to filtered_logs
        end if
        previous_end_time = end_time
    end if

end for
Sort filtered_logs by start_time
if operations is empty then
    Terminate with error: "No operation found"
end if
```

Algorithm Dispatch Elementary Actions to the PDDL Knowledge Base - P2 Dispatcher - (Part 2)

```
for each (op_name, job_id, resource, start_time, end_time) in operations do
    Retrieve actions:
    actions = operations_elementaires[job_id]["operations"][op_name.replace("_CO","")]
    if "pick" in actions or "place" in actions then
        tool_name = "tool_" + lowercase(op_name)
        Add tool_name to tools
        Add op_name to operations_requiring_tools
    end if
    if "move_to" in actions then
        move_to_name = "move_to_" + lowercase(op_name)
        Add move_to_name to move_to_operations
        loc_from = "loc_" + location_counter
        location_counter = location_counter + 1
        loc_to = "loc_" + location_counter
        location_counter = location_counter + 1
        Add (move_to_name, loc_from, loc_to) to locations
        op_locations[op_name] = loc_to
    else
        op_locations[op_name] = "loc_workstation"
    end if
end for
Generate domain.pddl (types, predicates)
if tools not empty then
    Add (holding), (tool_at), (can_operate)
end if
if a wait action is needed then
    Add (wait_done)
end if
for each op in operations_requiring_tools do
    Add (pick_op_done), (place_op_done)
end for
Add move_to_workstation action
```

Algorithm Dispatch Elementary Actions to the PDDL Knowledge Base - P2 Dispatcher - (Part 3)

```
for each (op_name, job_id, resource, start_time, end_time) in filtered_logs do
    if op_name = "wait" then
        Generate wait action
        previous_op_name = op_name
        continue
    end if
    Retrieve actions for op_name
    if "move_to" in actions then
        Generate corresponding move_to action
        previous_op_name = op_name
        continue
    end if
    if "pick" in actions then
        Generate pick_op_name action
        previous_op_name = op_name
    end if
    if "place" in actions then
        Generate place_op_name action
        previous_op_name = op_name
    end if
end for
Generate problem.pddl (objects, initial state)
if tools not empty then
    Add tool objects
end if
Add loc_base, loc_workstation, dynamic locations
Add agent_r
if operations_requiring_tools not empty then
    Add these operations
end if
Define initial state:
for each tool in tools do
    (tool_at tool loc_workstation)
end for
(at agent_r loc_base)
for each op in operations_requiring_tools do
    (can_operate tool_op op)
end for
Define goal:
for i = size(filtered_logs)-1 down to 0 do
    op_name = filtered_logs[i][0]
    if op_name not "wait" then
        Retrieve actions
        if "place" in actions and op_name in operations_requiring_tools then
            Goal: (place_op_name_done)
            break
        else if "move_to" in actions then
            loc = op_locations[op_name]
            Goal: (at agent_r loc)
            break
        end if
    end if
end for
return domain.pddl, problem.pddl
```