Trabajo Final PlanSys2
Envidio33

Alberto León Luengo
Jorge Martín Mínguez
Toni Marí Marí
Luis Moreno García

PDDL
(Tipos y Predicados)

Domain.pddl

```
(:types
  hall table bookshelf - location
  book miscellaneus - prop
  visitor
  robot
)
(:predicates
  (robot_at ?r - robot ?l - location)
  (object_at ?o - prop ?l - location)
  (visitor_at ?v - visitor ?l - location)
  (noise_at ?l - location)
  (connected ?l1 ?l2 - location)

  (gripper_free ?r - robot)
  (holding ?r - robot ?o - prop)
  (robot_not_busy ?r - robot)

  (solved ?m - miscellaneus)
  (book_found ?b - book ?l - bookshelf)
  (quiet ?l - location)
)
```

PDDL (Acciones Durativas)

Domain.pddl

```
(:durative-action move
  :parameters (?r - robot ?from ?to - location)
  :duration (= ?duration 2)
  :condition
    (and
      (at start (robot_at ?r ?from))
      (over all (connected ?from ?to))
      (at start (robot_not_busy ?r))
    )
  :effect
    (and
      (at start (not (robot_not_busy ?r)))
      (at start (not (robot_at ?r ?from)))
      (at end (robot_at ?r ?to))
      (at end (robot_not_busy ?r))
    )
)
```

Domain.pddl

```
(:durative-action shut_up
  :parameters (?r - robot ?l - location)
  :duration (= ?duration 2)
  :condition
    (and
      (over all (robot_at ?r ?l))
      (at start (noise_at ?l))
      (at start (robot_not_busy ?r))
    )
  :effect
    (and
      (at start (not (robot_not_busy ?r)))
      (at start (not(noise_at ?l)))
      (at end (quiet ?l))
      (at end (robot_not_busy ?r))
    )
)
```

Domain.pddl

```
(:durative-action search_book
  :parameters (?r - robot ?b - book ?l - bookshelf)
  :duration (= ?duration 2)
  :condition
    (and
      (over all (robot_at ?r ?l))
      (over all (object_at ?b ?l))
      (at start (gripper_free ?r))
      (at start (robot_not_busy ?r))
    )
  :effect
    (and
      (at start (not (robot_not_busy ?r)))
      (at end (book_found ?b ?l))
      (at end (robot_not_busy ?r))
    )
)
```

Behavior Tree

ShutUp.xml

```xml
<root BTCPP_format="4" main_tree_to_execute = "MainTree" >
    <BehaviorTree ID="MainTree">
        <Sequence name="root_sequence">
            <StandUp name="stand_up"/>
            <SayShhh name="say_shhh"/>
            <StandDown name="stand_down"/>
        </Sequence>
    </BehaviorTree>
</root>
```

SearchBook.xml

```xml
<root BTCPP_format="4" main_tree_to_execute="MainTree">
    <BehaviorTree ID="MainTree">
        <Sequence name="root_sequence">
            <Move name="move" goal="{arg3}"/>
            <Search name="Search" book="{arg2}"/>
        </Sequence>
    </BehaviorTree>
</root>
```

Controller Node

```cpp
bool init()
{
  domain_expert_ = std::make_shared<plansys2::DomainExpertClient>();
  planner_client_ = std::make_shared<plansys2::PlannerClient>();
  problem_expert_ = std::make_shared<plansys2::ProblemExpertClient>();
  executor_client_ = std::make_shared<plansys2::ExecutorClient>();

  init_knowledge();

  auto domain = domain_expert_->getDomain();
  auto problem = problem_expert_->getProblem();
  auto plan = planner_client_->getPlan(domain, problem);

  if (!plan.has_value()) {
    std::cout << "Could not find plan to reach goal " <<
      parser::pddl::toString(problem_expert_->getGoal()) << std::endl;
    return false;
  }

  if (!executor_client_->start_plan_execution(plan.value())) {
    RCLCPP_ERROR(get_logger(), "Error starting a new plan (first)");
  }

  return true;
}
```

```cpp
void step()
{
  if (!executor_client_->execute_and_check_plan()) {
    auto result = executor_client_->getResult();

    if (result.value().success) {
      RCLCPP_INFO(get_logger(), "Plan succesfully finished");
    } else {
      RCLCPP_ERROR(get_logger(), "Plan finished with error");
    }
  }
}
```

Plugin

PASO 1: Mueve la carpeta `plugin/plansys2_optic_plan_solver` que se encuentra en este paquete a la carpeta `ros2_planning_system`.

PASO 2: La estructura de la carpeta `ros2_planning_system` debe ser la mostrada a continuación.

```
> plansys2_bringup
> plansys2_bt_actions
> plansys2_core
> plansys2_docs
> plansys2_domain_expert
> plansys2_executor
> plansys2_lifecycle_manager
> plansys2_msgs
> plansys2_optic_plan_solver
> plansys2_pddl_parser
> plansys2_planner
> plansys2_popf_plan_solver
> plansys2_problem_expert
> plansys2_support_py
> plansys2_terminal
> plansys2_tests
> plansys2_tools
.gitignore
CODE_OF_CONDUCT.md
codecov.yaml
CONTRIBUTING.md
dependency_repos.repos
LICENSE
```

ros2_planning_system/
plansys2_bringup/params/
plansys2_params.yaml

```yaml
planner:
  ros__parameters:
    plan_solver_plugins: ["OPTIC"]
    POPF:
      plugin: "plansys2/POPFPlanSolver"
    TFD:
      plugin: "plansys2/TFDPlanSolver"
    OPTIC:
      plugin: "plansys2/OPTICPlanSolver"
executor:
  ros__parameters:
    bt_builder_plugin: "SimpleBTBuilder"
```

Action Nodes

```cpp
BT::NodeStatus
Move::on_tick()
{
  if (status() == BT::NodeStatus::IDLE) {
    rclcpp_lifecycle::LifecycleNode::SharedPtr node;
    if (!config().blackboard->get("node", node)) {
      RCLCPP_ERROR(node_->get_logger(), "Failed to get 'node' from the blackboard");
    }

    std::string goal;
    getInput<std::string>("goal", goal);

    geometry_msgs::msg::Pose2D pose2nav;
    if (waypoints_.find(goal) != waypoints_.end()) {
      pose2nav = waypoints_[goal];
    } else {
      std::cerr << "No coordinate for waypoint [" << goal << "]" << std::endl;
    }

    geometry_msgs::msg::PoseStamped goal_pos;

    goal_pos.header.frame_id = "map";
    goal_pos.header.stamp = node->now();
    goal_pos.pose.position.x = pose2nav.x;
    goal_pos.pose.position.y = pose2nav.y;
    goal_pos.pose.position.z = 0;
    goal_pos.pose.orientation = tf2::toMsg(tf2::Quaternion({0.0, 0.0, 1.0}, pose2nav.theta));

    goal_.pose = goal_pos;
  }

  return BT::NodeStatus::RUNNING;
}
```

```cpp
BT::NodeStatus
SayShhh::tick()
{
  std::cout << "SayShhh tick " << counter_ << std::endl;

  if (counter_++ < 5) {
    return BT::NodeStatus::RUNNING;
  } else {
    counter_ = 0;
    return BT::NodeStatus::SUCCESS;
  }
}
```

FIN

Gracias por vuestra atención