PNP-ROS

SAPIENZA
UNIVERSITÀ DI ROMA

Luca Iocchi, Fabio Previtali
Dipartimento di Ingegneria Informatica
Automatica e Gestionale

PNP-ROS

- Bridge between PNP and ROS
- Allows execution of PNP under ROS using the actionlib module
- Defines a generic PNPAction and an ActionClient for PNPActions
- Defines a client service PNPConditionEval to evaluate conditions

PNP-ROS download

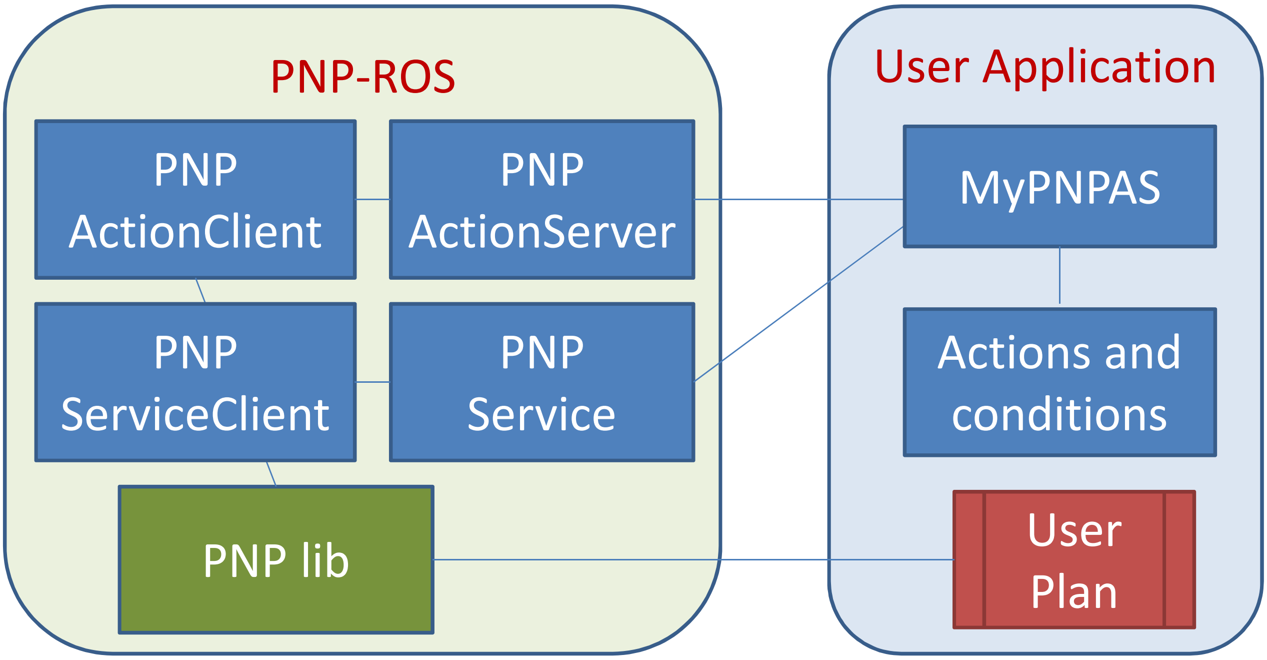
Download and follow instructions from

https://github.com/iocchi/PetriNetPlans

Use stable version v1.1 (branch v1.1)

```
$ git clone -b v1.1 https://github.com/iocchi/PetriNetPlans.git
```

PNP-ROS



PNP-ROS

User development:

1. implement actions and conditions – Writing a PNPActionServer
2. write plans – Using Jarp

PNPActionServer

```
class PNPActionServer
{
public:
   PNPActionServer();
   ~PNPActionServer();
   void start();
   // To be provided by actual implementation
   virtual void actionExecutionThread(string action_name,
                                      string action_params, bool *run);
   virtual int evalCondition(string condition); // 1: true, 0: false; -1:unknown
}
```

PNPActionServer

```
class PNPActionServer
{
public:
   …
   // For registering action functions
   void register_action(string actionname, action_fn_t actionfn);
   void register_MRaction(string actionname, MRaction_fn_t actionfn);
                  // multi-robot version

}
```

MyPNPActionServer (v.1)

```
class MyPNPActionServer : public PNPActionServer
{
public:
    // Actual implementation
    void actionExecutionThread(string action_name, string
    action_params, bool *run)
    { … }

    bool evalCondition(string condition)
    { … }
}
```

MyPNPActionServer (v.1)

void **actionExecutionThread**(string action_name, string action_params, bool *run)

- Executes the action action_name with parameters action_params
- until run is true. If run becomes false during the execution, it must be interrupted.
- Blocking function. It returns when the action is finished.

bool **evalCondition**(string condition)

- Return the boolean value of condition

MyPNPActionServer (v.2)

```
#Include "MyActions.h"

class MyPNPActionServer : public PNPActionServer
{
   MyPNPActionServer() : PNPActionServer() {
        register_action("init",&init);
        ….
        event_pub = // asynchronous conditions
        handle.advertise<std_msgs::String>("PNPConditionEvent", 10);
   }
}
```

MyPNPActionServer (v.2)

For a complete example, see

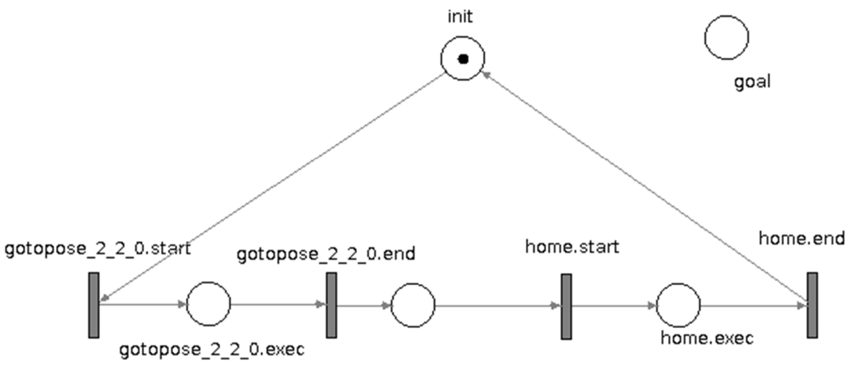
PNPros/example/rp_action/src/nodes/MyPNPAS.cpp

How to write plans

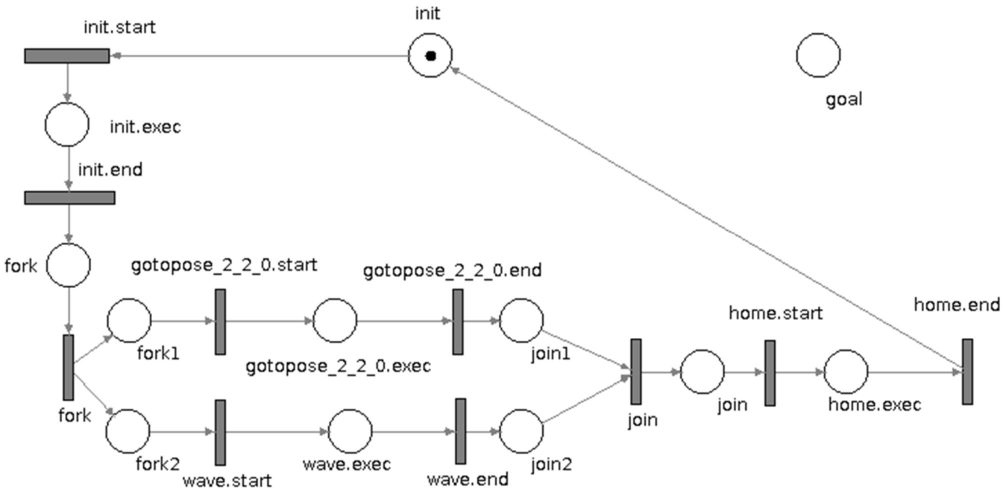
Jarp is a graphical interface to write Petri Nets, that can be used to write PNPs

Available in the Jarp directory of PetriNetPlans repository

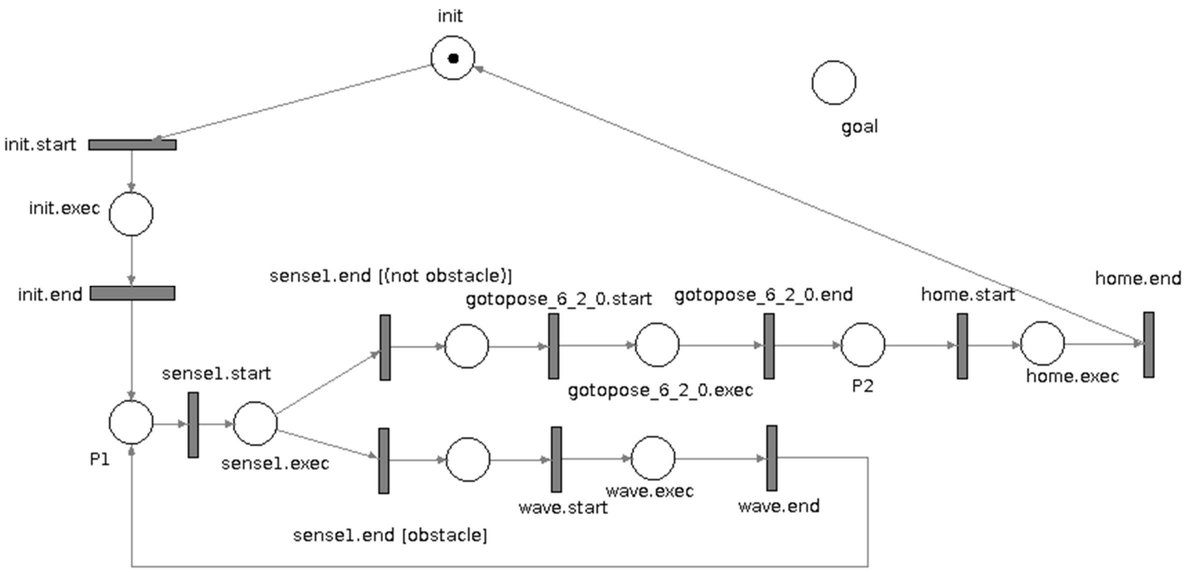
Plan 1: sequence and loop



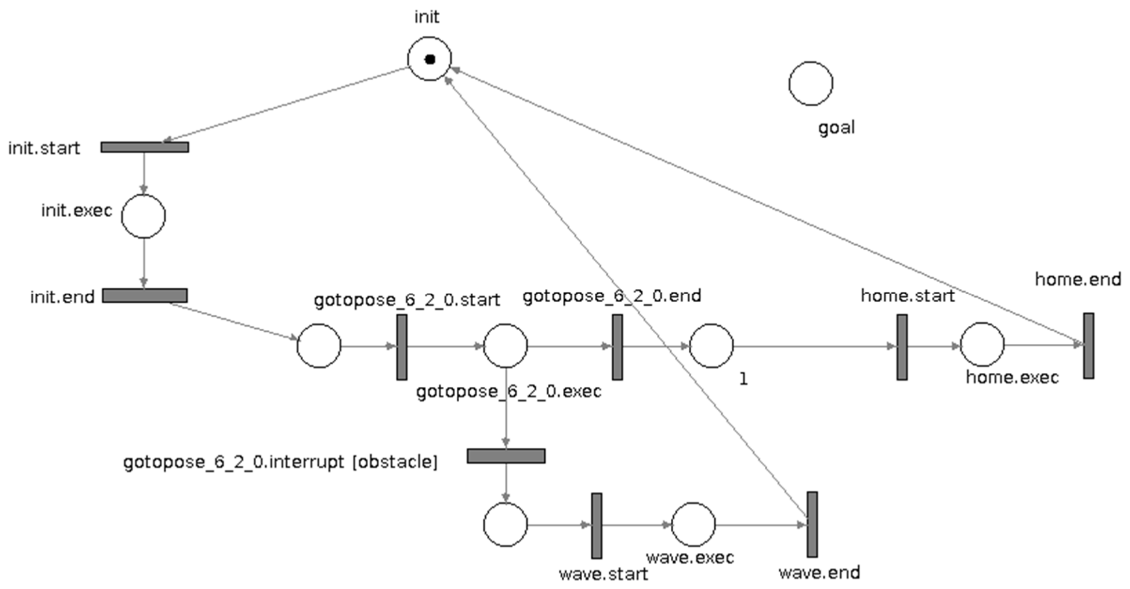
Plan 2: fork and join



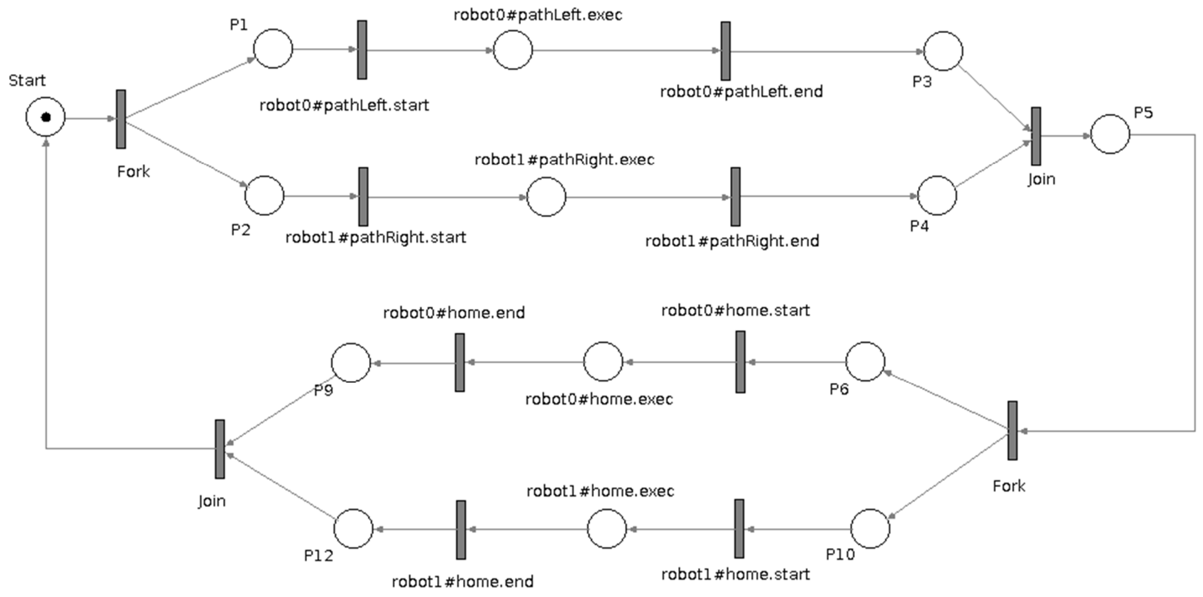
Plan 3: sensing



Plan 4: interrupt



Plan 5: multi robot



PNPros Execution

Execute the script

```
PNPros/example/rp_action/scripts$ ./run-dis-B1-plan.sh
```

Homework

1. Implement the action 'Turn360' (that makes the robot turn on itself 360 degrees)
2. Implement a condition 'closeToHome' that determines if the robot is within 5 meters from the home position (which is (2,2)) – use getRobotPose function to read the current pose of the robot
3. Modify the interrupt for obtaining the following behavior:
   - the robot must patrol both the corridors
   - if there is an obstacle in front of the robot, then:
     - if the robot close to home, it must get back there
     - otherwise, make the Turn360 action, until the obstacle moves away, then continue towards the previous goal