Package for LTL motion and task planning

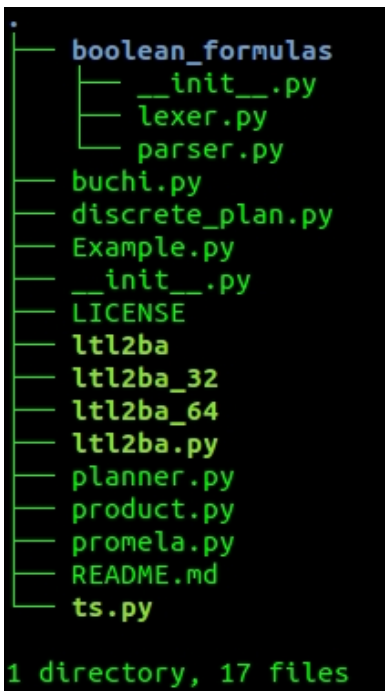
Meng Guo

November 13, 2015

General info of P_MAS_TG

- Pure Python 2.7 based. Size: 0.3MB[1]

- Structure (1 directory, 17 files):



- Dependent external packages:
  - `ply`, for parsing LTL formulas
  - `networkx`, for graph construction and search
  - `ltl2ba`, executable for LTL2BA[2] (needs to be compiled under your OS)

[1] `https://github.com/MengGuo/P_MAS_TG.git`

[2] `http://www.lsv.ens-cachan.fr/%7Egastin/ltl2ba/download.php`

Details (1)

- `ts.py`: construct or modify FTS as `networkx.digraph`
  - `.add_node()`, `.add_edge()`, `.set_initial()`...
  - Allows both motion and action FTS, and composition
  - Update FTS given new info
- `buchi.py`: construct Büchi from LTL formulas as `networkx.digraph`
  - `.buchi_from_ltl()`...
  - Allows partially-infeasible task, soft and hard tasks[3]
  - Based on `promela.py` and folder `boolean_formulas`
  - Execute `ltl2ba` $\longrightarrow$ parse results $\longrightarrow$ construct Büchi graph

[3] IJRR15

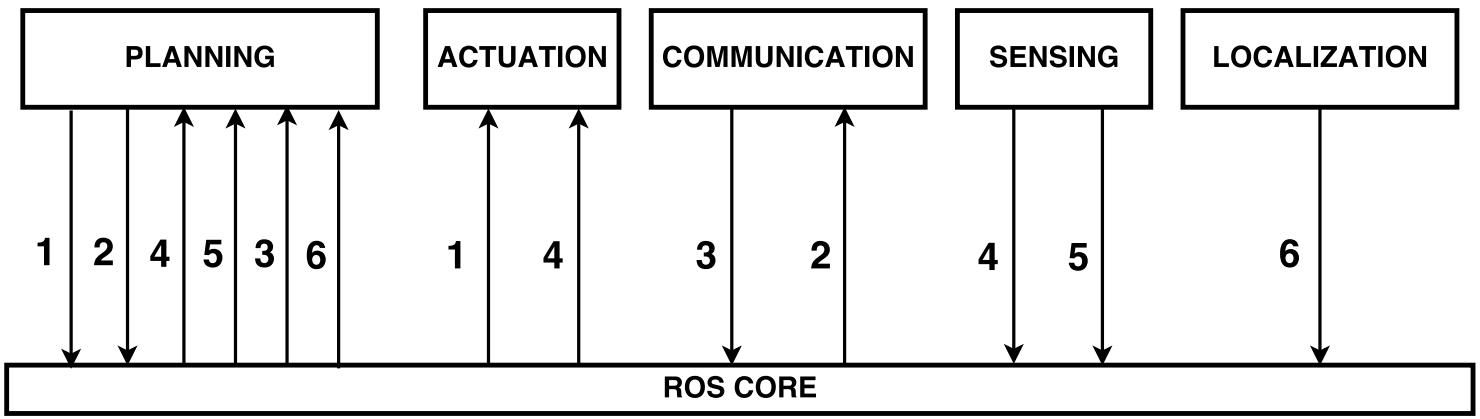
Details (2)

- `product.py`: construct product automaton as `networkx.digraph`
  - `.build_full()`, `.fly_successors_iter()`...
  - Allows static and on-the-fly construction
  - Product update after FTS update
  - Class for accepting run (good for projection and prefix-suffix structure)

- `discrete_plan.py`: find accepting runs of the product automaton
  - `.dijkstra_plan_optimal()`, `validate_and_revise()`...
  - Plan synthesis and revision algorithms
  - More search algorithm can be added, e.g., DFS, BFS, A*

Details (3)

- `planner.py`: contains the `ltl_planner` class
  - Highest class as the planning node, e.g., `.product`, `.trace`, `.index`, `.segment`...
  - Contain functions, `.replan()`, `.update()`
  - Interacts with other nodes as below



(1) Stand-alone plan synthesis

Run `Examples/case.py`

- Motion planning.
- Motion and action planning.
- Infeasible task.
- Soft and hard task.

(2) With ROS

Show `Examples/youbot`

- Initialize agent motion and action FTS in `agent_init.py`
- Run ROS `agent_planner.py` with agent ID as extra argument[4]
- It synthesizes `.plan`
- It sends `next_activity` message
- It receives `confirmation` and `knowledge_update` messages

[4] **Do NOT upgrade to Ubuntu 15.10 LTS**

(3) Multi-agent systems

- Fully decentralized, see `Examples/youbot`
  - Local planner for each agent
  - Message passing medium like ROS
- Semi-decentralized, see `Examples/multi`[5]
  - Initialize agent model and planner in `agents_init.py`
  - All agent planners accessible by each agent in `agents_planner.py`
  - Agents run in sequence, not strictly parallel
  - No communication module needed
  - Add additional functions to `ltl_planner()` to suit your purpose

[5] CASE15

Extensions

(1) With Aeroworks

- GUI to define FTS and task
  - Waypoints and properties
  - Reference trajectory
  - Task formula
- Collaborative manipulation
- Relative leader-follower formation

(2) To timed temporal logic

- "FTS $\longrightarrow$ product $\longrightarrow$ structure" still useful
- Need new package for Metric Interval Temporal Logic (MITL)
  - Parsing rules
  - MITL2BA?
  - Product automaton

Summary

- Structure
- Applications
- Extensions

Questions?