Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League

José Carlos González
Ángel García-Olaya
Fernando Fernández

Planning and Learning Group

April 15th, 2020

Computer Science Department
• Autonomous robotics is complex
  ▪ Many types of sensors and actuators
  ▪ Deliberation to behave coherently

• Deliberation is complex
  ▪ Now feasible for real systems

• Use cases must be defined
  ▪ How to really implement them?

• Architecture for coordination
  [Kortenkamp et al. 2008]
Automated-planning technique

**PDDL**

**Domain (action)**

(:action pickup
 :parameters (?ob)
 :precondition (and
 (clear ?ob)
 (on-table ?ob)
 (arm-empty))
 :effect (and
 (holding ?ob)
 (not (clear ?ob))
 (not (on-table ?ob))
 (not (arm-empty))))

**Problem**

(:init
 (on-table a) (on-table b)
 (clear a) (clear b)
 (arm-empty))
 (:goal (and (on a b))))

**Plan**

1: pickup(a)
2: stack(a,b)

**Classical**

- Simplest
- Fastest

**Probabilistic**

- Explicit probabilities

**Temporal**

- Explicit durations
- Overlapping actions
- Overall and at-end

[Introduction]

Mlaras architecture

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League

[Ghallab et al. 2004]
Mlaras architecture

- Multi-Layered ARchitecture for Autonomous Systems
  [González 2020]
- Focused on classical AP
  - Stochastic: monitoring and replanning
  - Temporal: condition annotations
  - Hierarchical: layered deliberation
- Explicit abstraction conversions
- Declarative use case definition

Introduction
Mlaras architecture
Logistics competition
Multi-Layered Multi-Robot Control Architecture
for the Robocup Logistics League
Mlaras modules

- Deliberation layer overview

Diagram showing the flow between different modules:
- Goal Converter
- State Converter
- Action Converter
- Monitoring
- Search

States and actions such as Goal+, State_, Action, State_, and State_ are connected with arrows indicating the flow and interaction between these modules.
Declarative configuration

Mlaras modules

- Goal Converter
  - Goal relations
  - Action\(_H\)
  - Stop
  - State\(_H\)
  - Idle

- State Converter
  - State generalizations
  - State\(_L\)
  - Idle
  - Action\(_L\)
  - Stop

- Action Converter

- Executive
  - Agent state

- Search
  - External solver

- Monitoring
  - Model state

- Domain
- Problem template
- Action decompositions
- Durative conditions

Introduction
Mlaras architecture
Logistics competition

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
RoboCup logistics simulation league

• **Goal**
  - Product building orders
  - More rings for more score

• **Robots**
  - 2 teams of 3 robots
  - Should cooperate

• **Products**
  - Base (3 colors)
  - From 0 to 3 rings (4 colors)
  - Cap (2 colors)

[Niemueller et al. 2015]
RoboCup logistics simulation league

- Stations
  - Base, ring, cap, delivery
  - Errors break them
  - Require bases to work

- Orders
  - Supplied by the referee
  - Product, gate and time window
  - Appear randomly
  - Completion gives score
  - Only 15 minutes

Multi-Layered Multi-Robot Control Architecture for the RoboCup Logistics League
Mlaras logistics for RC logistics

- Parts replaced
  - Deliberation
  - Abstraction conversions

- New PDDL domain
  - Based on classical planning
    - Centralized plan for the 3 robots
      [Borrajo et al. 2019]

- Opportunities management
  - Rescheduling for new orders

1. (GO_TO R1 C-CS1 IN START_AREA)
2. (GO_TO R2 C-BS OUT START_AREA)
3. (GO_TO R3 C-CS1 OUT START_AREA)
4. (RETRIEVE_BASE_SHELF R1 C-CS1 IN B_TR)
5. (RETRIEVE_BASE R2 C-BS OUT B_BL)
6. (FEED_CAP_STATION R1 R3 C-CS1)
7. (GO_TO R2 C-RS1 IN C_BS)
8. (RETRIEVE_BASE_SHELF R1 C-CS1 IN B_TR)
9. (FEED_RING_STATION R2 C-RS1 IN B_BL)
10. (GO_TO R3 C-RS2 IN C-CS1)
11. (GO_TO R2 C-BS OUT C-RS1)
12. (GO_TO R1 C-CS2 IN C-CS1)
13. (FEED_RING_STATION R3 C-RS2 IN B_TR)

...
Mlaras logistics layer overview

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League

Level 0

Reactor

State_0

Action_0

Level 1

Robot actions

State_1

Action_1

Level 2

Order schedule

State_2

Action_2

Robots

Monitoring

Orders

Referee
Results

• Planning
  ▪ Planning time of 10 seconds is enough
  ▪ The three agents execute orders in parallel
  ▪ In spite of using classical planning approach

• Scores
  ▪ Over 170 points if there are no obstructions
    – Last winner 139 points, runner-up 32 points

• Full declarative configuration
Conclusions

- Mlaras control architecture
  - Eases the development of autonomous systems
  - Declarative languages to ease use case refining

- Planning and Execution for Logistics in Simulation
  - Classical planning approach
  - Stochastic, temporal, hierarchical and multiagent aspects
  - Competitive results

- Future work
  - Better multiagent parallelization
  - Improve the high-level scheduler
References


Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League

José Carlos González
Ángel García-Olaya
Fernando Fernández

Planning and Learning Group

Thank you for your attention
Problems of SoA architectures

• Lack of guidelines and standards
• Difficult to reuse previous works
• Use cases are hardcoded by developers
  ▪ However, they must be defined by end users
• Hardcoded abstraction conversions
• Complex and slow deliberation models
• Lack of multilayer deliberation support

All them slow down the development and advancement of autonomous systems
Control strategies

• Procedural control
  ▪ Behavioral trees
  ▪ Tree sets can model use cases
  ▪ Action decompositions save deliberation time

• Deliberation
  ▪ Decomposed deliberation
  ▪ Higher layers: deliberative
  ▪ Lower layers: reactive
  ▪ Several simpler problems are easier
Abstraction layers

• **High-level**
  - High states to deliberate with (PDDL)
  - High actions to define behaviors

• **Low-level**
  - Low states with data from the sensors
  - Low actions are instructions for the robot

---

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
Managing temporal aspects

• Actions have duration
  ▪ Can be unknown
  ▪ Interruption in the middle of their execution

• Overall and finish-when annotations
  ▪ Similar to temporal planning without overlapping actions
  ▪ Controlled by Mlaras, not by the planner
State separation

• Model state
  ▪ Minimum predicate set to deliberate with in Search
  ▪ Corresponds with $State_L$ and the $State_M$

• Agent state
  ▪ Predicates used only to control abstraction conversions
  ▪ Not present in any state (optionally in $State_H$)

• Layer accesses
  ▪ Read only: Model state, action parameters
  ▪ Read and write: Agent state
  ▪ Variable resolution order: parameter $\rightarrow$ agent $\rightarrow$ model

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
Durative conditions

- Order is important: First match is selected

Medium: `deliver-object(object, person, location_a, location_b),
?object is coffee`
Priority: 10
Cooldown: 360000
Overall: `?held_object is ?object`
Opportunity: `?nearest_person is ?person`

Medium: `introduce-session(pres, robot, pause, 1st_name, 2nd_name)`
Overall: `?person and not ?battery and not ?help and not ?stop and not ?stop_session`
Finish-when: `not ?is_speaking`

Medium: `RETRIEVE_BASE (robot, station, side, color)`
Finish-when: `?agent_status == 3 and
?agent_last_low_times >= ?pelea/last_low_times`
Action decompositions

- Low-level actions of each set are executed in parallel

High: say(speech, behavior), ?behavior is show_video
High: say(speech, behavior), ?behavior is show_tutorial
Lows: say(?speech)
show_video(?speech)
?pauseatend(600)

High: say(speech, behavior)
Lows: say(?speech)

High: finish-pose(), ?counter_skipped_pose < 3
Lows: print("FINISH-POSE (skipped)")
executeAnimation(green_eyes)
Lows: say(speech_clue)
Lows: executeAnimation(blinking)
?set(counter_failed_pose, 0)
?increase(counter_skipped_pose, 1)
Action decompositions

- High-level set = Branch of a behavioral tree node
- Powerful and easy tool to refine use cases
- Low-level actions depend on the lower level layer catalog

High: `finish-exercise-mirror()`
High: `finish-exercise-simon(mode), ?mode==classic_finish`
Lows: `print("FINISH-EXERCISE")`
Lows: `allowAutonomousMovements(true)`
Lows: `executeAnimation(blinking)`
Lows: `executeAnimation(initFull)`
Lows: `say(speech_success)`
Lows: `pauseatend(200)`
Lows: `say("We have finished this exercise.")`
Lows: `executeAnimation(blinking)`
Lows: `say(speech_encouragement)`
Lows: `pauseatend(1000)`
State generalizations

- Order is not important, all conditions are evaluated
- Agent state is only written here

If: True
   add(checkPoseResult ?checkPoseResult, model)
   add(exerciseTotalPoses ?exerciseTotalPoses, model)

If: not ?person
   If: ?emergency_situation
      delete(can_continue, model)

If: ?checkPoseResult is OK and ?ready
   increase(performedPoses 1, agent)
Goal relations

- All \( \text{Action}_H \) received in parallel are checked sequentially.

High: execute_order(id, base, ring1, ring2, ring3, cap, gate),
      \(?\text{ring1 is void}\) and \(?\text{ring2 is void}\) and \(?\text{ring3 is void}\)

Goal: add(product_piece ?id zero ?base, goals)
      add(product_cap ?id ?cap, goals)
      add(product_gate/~/?id ?gate, agents)
      add(delivered ?id, goals)

High: execute_order(id, base, ring1, ring2, ring3, cap, gate),
      \(?\text{ring3 is void}\)

Goal: add(product_piece ?id zero ?base, goals)
      add(product_piece ?id one ?ring1, goals)
      add(product_piece ?id two ?ring2, goals)
      add(product_cap ?id ?cap, goals)
      add(product_gate/~/?id ?gate, agents, goals)
      add(delivered ?id, goals)
Opportunities management

Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League
Mlaras comparison

- Mlaras eases the development of autonomous systems
- Evaluations in 3 real and 1 simulated environments

<table>
<thead>
<tr>
<th>Deliberation</th>
<th>Stochastic</th>
<th>Temporal</th>
<th>Declarative</th>
<th>Multilayer</th>
<th>Middleware</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAAS¹</td>
<td>Cust. AP</td>
<td>Ad hoc</td>
<td>Cust. AP</td>
<td>Ad hoc</td>
<td>✓</td>
</tr>
<tr>
<td>T-REX²</td>
<td>AP</td>
<td>Replan</td>
<td>Timelines</td>
<td>Partial</td>
<td>✓</td>
</tr>
<tr>
<td>PELEA³</td>
<td>AP</td>
<td>Replan</td>
<td>Temp. AP</td>
<td>Partial</td>
<td>✗</td>
</tr>
<tr>
<td>ROSPlan⁴</td>
<td>AP</td>
<td>Replan</td>
<td>Temp. AP</td>
<td>Partial</td>
<td>✗</td>
</tr>
<tr>
<td>CORTEX⁵</td>
<td>Ad hoc</td>
<td>Ad hoc</td>
<td>Ad hoc</td>
<td>Ad hoc</td>
<td>✗</td>
</tr>
<tr>
<td>Mlaras⁶</td>
<td>AP</td>
<td>Replan</td>
<td>Classic AP</td>
<td>Use case</td>
<td>✓</td>
</tr>
</tbody>
</table>

1. [Alami et al. 1998] 4. [Cashmore et al. 2015]
Comparison references


Multi-Layered Multi-Robot Control Architecture for the Robocup Logistics League