CLIPS Executive – Fundamentals

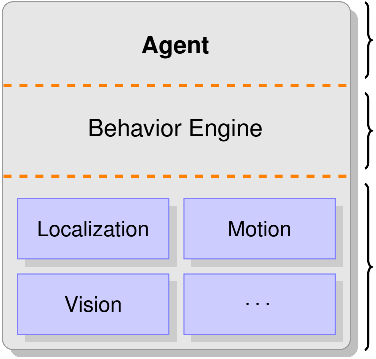
Lab Course Winter Term 2021/2022

Till Hofmann, Tarik Viehmann

KB SG Knowledge-Based Systems Group | RWTH AACHEN UNIVERSITY

October 18, 2021

Behavioral Architecture



Deliberation
Decision making/planning

Reactive Behaviors
Skill execution/monitoring

Components
Actuators/Sensors
Data Processing

Goal Reasoning with the CLIPS Executive

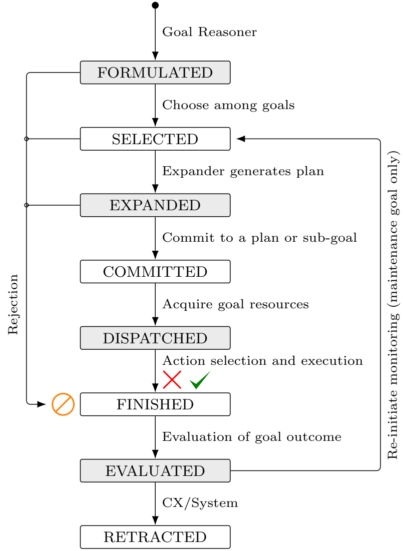
- Typically: reason about actions, but goal is fixed
- *Goal reasoning*:
  - Explicitly represent goals
  - Continually reason about goals
  - Dynamically adjust and prioritize goals
  - Model flow along *goal life cycle*
- Reason about *what* to accomplish, only then *how* to accomplish it

Goal Reasoning with the CLIPS Executive

- Typically: reason about actions, but goal is fixed
- *Goal reasoning*:
  - Explicitly represent goals
  - Continually reason about goals
  - Dynamically adjust and prioritize goals
  - Model flow along *goal life cycle*
- Reason about *what* to accomplish, only then *how* to accomplish it

- Explicitly represent **goals as first class object**
- Model **flow** along *goal lifecycle*
- Clearly define **components** to separate concerns
- Specify explicit *interfaces* in terms of *facts and flow*
- Make each component *exchangeable* (as far as possible)

Goal Lifecycle



[Niemueller et al., ICAPS 2019]

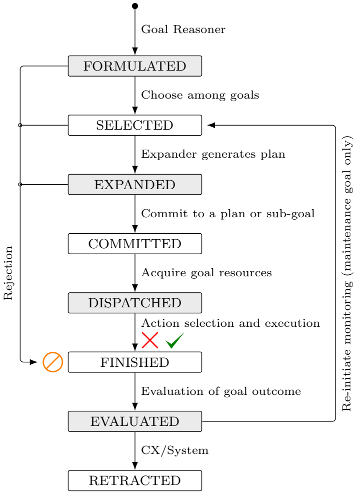
Components

Goal Reasoner
: Formulates and expands goals

Goal Expander
: Expands a goal into a plan,
e.g., with PDDL planning

Action Executor
: Executes a single action on the robot,
e.g., with the behavior engine

Monitor
: Monitors the execution of a plan,
adapts plan dynamically,
e.g., by retrying actions



World Model

- World model is a set of key-value pairs,
  e.g., `/mps/C-BS/state:  PREPARED`
- Contains perception and communication feedback
- Domain model updated during execution propagate

- Generic storage allows for exchangeable sync/storage
- Two-way communication world and domain model
- Separates reasoning (domain) model from information

Goal Reasoning

- Subset of what you can find in the literature
- Domain-specific set of reasoning rules
- Formulate and select goals
- “Source of goals”
- Aka mission controller, strategic reasoner, deliberation
- Formulates and selects goals, commits to plans

Input World model

Output Goal facts

```
    (goal (id X) (mode FORMULATED...)  ...)
```

Goal Expander

- Generate plan for selected goals
- More than one goal may be selected
- More than one expander may generate a plan (in principal)
- Can be library of procedures (similar to tasks and steps)
- Can be a planning system, e.g., PDDL-based

Input `(goal (id X) (mode SELECTED) ...)`

Output Switch goal to EXPANDED state

```
    (plan (id X-PLAN) (goal-id X)))
    (plan-action (id 1) (plan-id X-PLAN) (duration 2.0)
           (action-name foo) ...)
```

Action Selection and Execution

Action Selection/Plan Execution

- Currently: PDDL domain precondition verification
- For multiple goals/plans: select one out of many actions
- Generic and domain-specific variants (and mixes)

Input Domain model, plan actions

Output Plan actions marked executable

Action Selection and Execution

Action Selection/Plan Execution

- Currently: PDDL domain precondition verification
- For multiple goals/plans: select one out of many actions
- Generic and domain-specific variants (and mixes)

Input Domain model, plan actions

Output Plan actions marked executable

Action Execution

- Generic skill execution
- Map plan actions to skill strings via config

Input Executable action

Output Execute actions (per-action state machine)

Execution Monitoring

- Monitor plans and actions during execution
- Generic checks, e.g.,
  - no action executable anymore
  - expected effect does not coincide with sensing result
- Domain-specific checks, e.g.,
  - product lost while driving
  - heuristic to mark plan infeasible based on time

Input Goal and action state

Output Advice to goal reasoner, direct influence on goal (risky)

World Model Synchronization

- Parts of the world model are synchronized between all agents
  → shared world model
- Each robot runs a *MongoDB* instance
- Each MongoDB instance is part of a replica set

→ If one robot fails, other robots still have full (shared) world model

→ If a robot is re-inserted, it gets the world model from the other robots

Multi-Agent Task Coordination

- Basic mechanism: mutual exclusion with locks
- Part of the worlmodel is shared between the agents
- Coordination with three kinds of locking mechanisms:
  1. A goal may require a *resource*,
     which is assigned to the goal for the whole lifetime of the goal
  2. *Lock actions* acquire and release a mutex within a plan
  3. *Location locks* guarantee that no two robots drive to the same location

⇒ *Cooperative* and *Negotiated Distributed Planning*

ICAART 2021

The RCLL Agent Setup

Clips-Executive (CX)

- Goal reasoning framework
- Implemented as `fawkes` plugin
- Additional plugins to extend *features*
- Skeleton for agents written in `CLIPS`
- Agent = CX + custom domain, goal reasoning, plan execution and execution monitoring
- fawkes-robotino: application-specific configs/plugins and the RCLL agent
- 3 stage setup:
  1. Required features (user-defined)
  2. CX core files (static)
  3. CX specifics (user-defined)

Conclusion

- Explicit **goal** representation
- **Reason about goals**, not only about actions with one fixed goal
- **Goal lifecycle** to model program **flow**
- Exchangable **components** to separate concerns
- **Distributed Multi-Agent Reasoning**:
  - Synchronized world model
  - Coordination with **resource locks** and **lock actions**