

# **Εργασία παρουσίασης άρθρου**

**Logic programming for deliberative robotic task planning**

**Νικόλας Χελιώτης**

# Context

# Context

## **Autonomy**

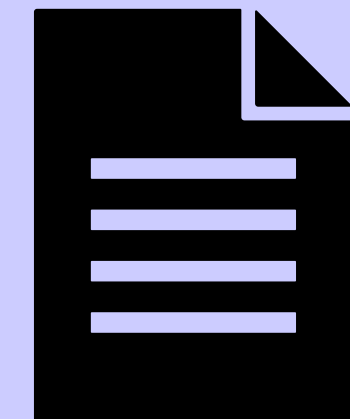
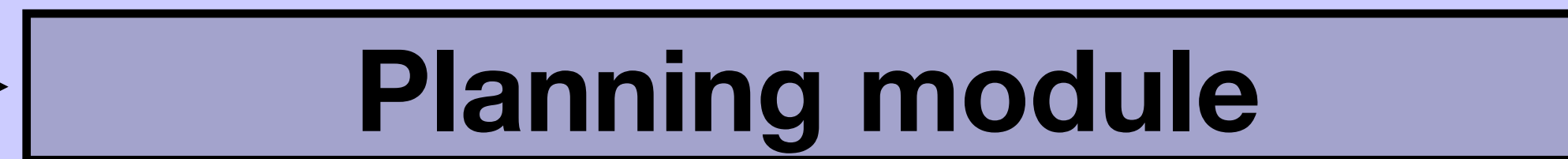
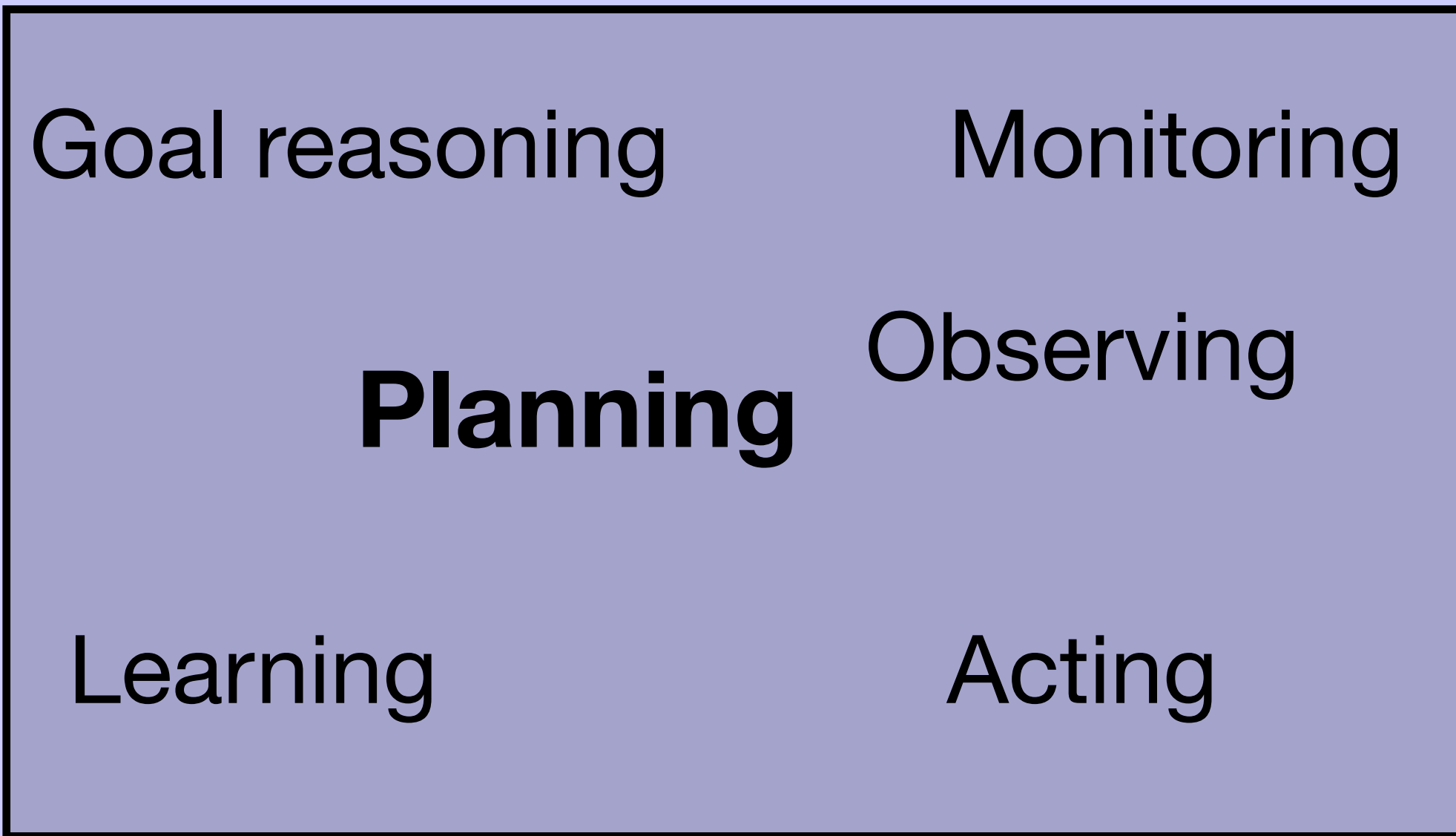
*“The ability to fulfill a mission without external supervision.”*

## **Deliberation**

*“The ability to make decisions which are motivated by reasoning on the available resources, the capabilities of the robot, the actual description of the environment and the given mission.”*

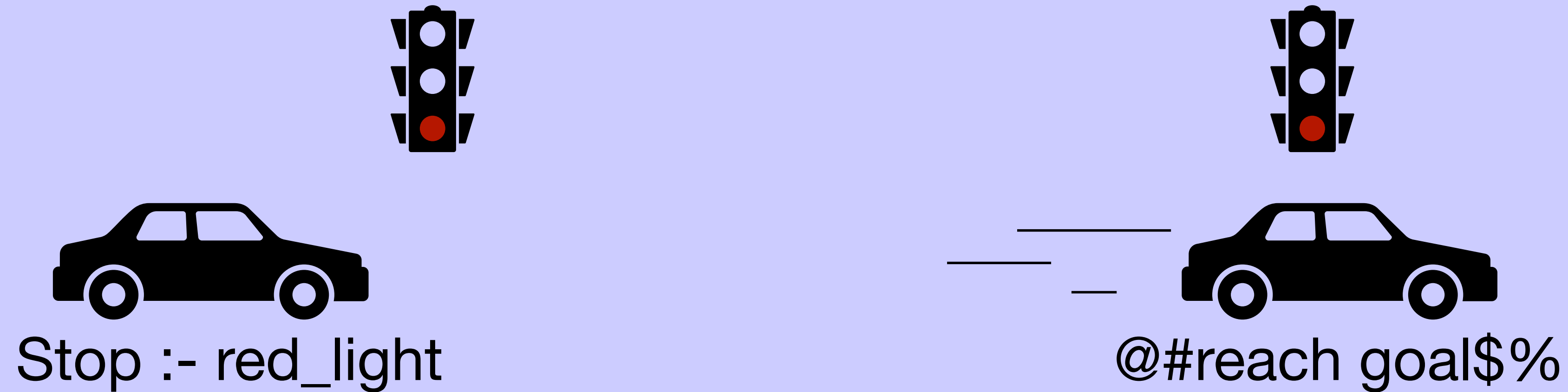
# Context

## Deliberation



# Context

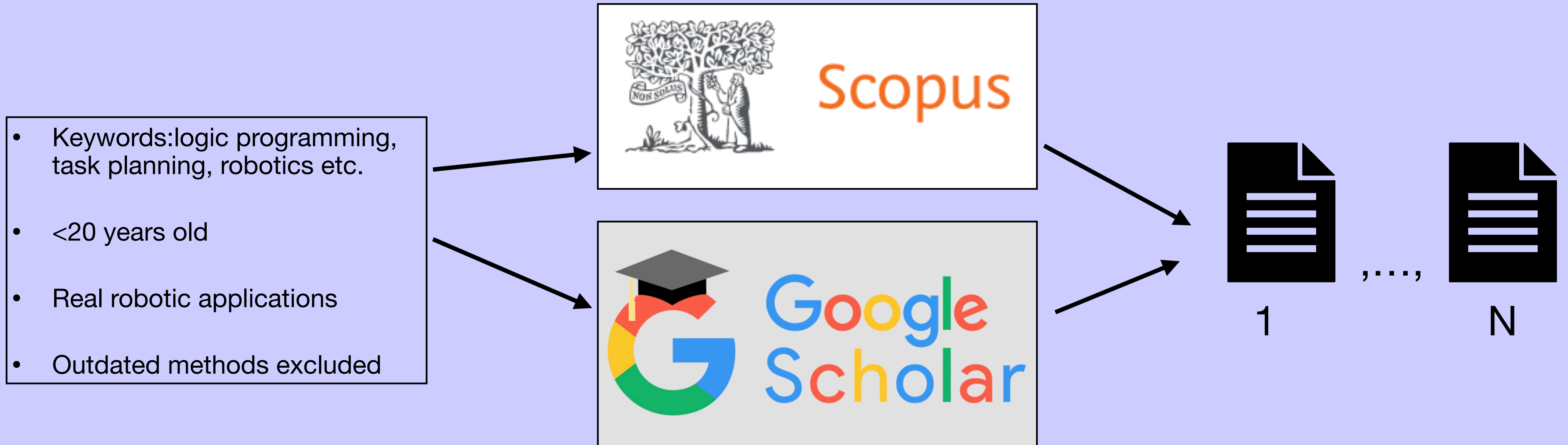
## *Why logic task planners?*



- Η αυτονομη συμπεριφορα ειναι interpretable απο τον χρηστη (input και output)
- Interpretable planning -> Ζητουμενο EU για υψηλου κινδυνου συστηματα.

# Context

*How was the paper created?*



# Taxonomy of planners

# Taxonomy of planners

**Domain-depedent vs Domain-independent** ✗

**Deterministic vs Non-Deterministic** ✗

**Classical planners, Temporal planners, Probabilistic planners** ✓

↓  
Deterministic environment

↓  
Time-sensitive tasks

↓  
Uncertainty and randomness



# Planning Domain Description Language (PPDL)

# PPDL vs Prolog

**-What is it?** A language to describe what is possible in a given problem.



- Why?**
- PDDL is useful for planning problems but it lacks reasoning capabilities.
  - If conditions change, PDDL-based planners must recompute everything.

# Criteria for logic framework selection

# Criteria for logic framework selection

1. **Expressivity** (How much can it describe?)

2. **Computational Efficiency** (How fast can it compute a plan?)

- Classical-> PSPACE-complete
- Temporal-> EXSPACE- complete
- Probabilistic -> Unpredictable

3. **Software Implementation** (Can it work with standard robotic software like ROS?)

- Clingo and Prolog widely supported

# Criteria for logic framework selection

## 4. **Support for open-world planning** (Can it work with unknown environments?)

-Prolog-based planners are great for querying external knowledge.

## 5. **Plan revision opportunity** (Can it update in real time if the situation changes?)

-Prolog based planners are non-monotonic meaning they can revise plans in real time.

-That is not the case for temporal planners.

# Standard logic programming

# Standard logic programming

## 1. Prolog based planners:

- Instead of searching for plans, they deduce for the next action.
- Dynamic capabilities if the environment changes (Non-monotonicity).
- Easier to debug.
- Represents knowledge in tree structure.

## 2. ASP- based planners:

- Creates all possible plans and picks the best one. Good for optimisation problems.
- Monotonic.

## 3. Other planners: CLP (numerical constraints), Hybrid (Mixed LP and ML).

# Temporal logic programming



# Temporal logic programming

-Sometimes, we need extra expressivity to describe conditions and relations.

Temporal logic solves this problem by introducing new operators (eventually, always, etc.) regarding time.

-Logic programming is timeless. It reasons about facts and rules without thinking about when things should happen.

- TL frameworks:**
1. **Linear Temporal Logic** (Single sequence of future events)
  2. **Computational Tree Logic** (Multiple possible futures)
  3. **Timed Automata** (Models real time systems)

# Probabilistic logic programming

# Probabilistic logic programming

- The problem with standard logic programming is that it assumes that everything is either true or false.
- Uncertainty is everywhere (Sensors can be noisy.95% prob shelf\_not\_empty)

## -PL frameworks:

- 1.**Problog** (extends Prolog with prob reasoning)
- 2.**LPAD** (allows multiple uncertain outcomes per rule)

Questions?

Thank you :)