# *Endomorphic metalanguage and abstract planning for real-time intent recognition*

**Antoine Gréa**

Lyon 1

**12020-01-30T14:00+01**
ISO-HE

- Directors
  - Samir Aknine
  - Lætitia Matignon
- Jury
  - Hamamache Kheddouci
  - Ivan Varzinczac

LIRIS

UNIVERSITÉ D'ARTOIS

- Reviewers
  - Eva Onainda
  - Damien Pellier

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

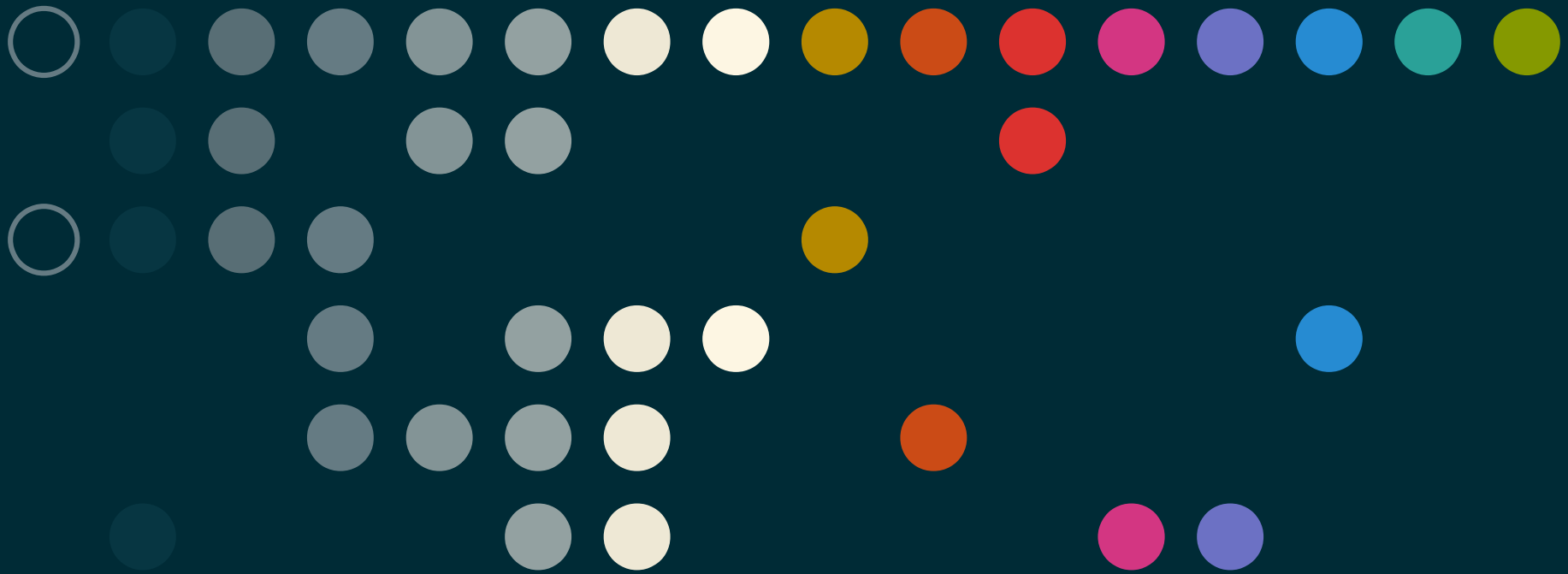
LIG

# *Endomorphic metalanguage and abstract planning for real-time intent recognition*

LIRIS Lyon 1 Antoine Gréa

# 1 Introduction



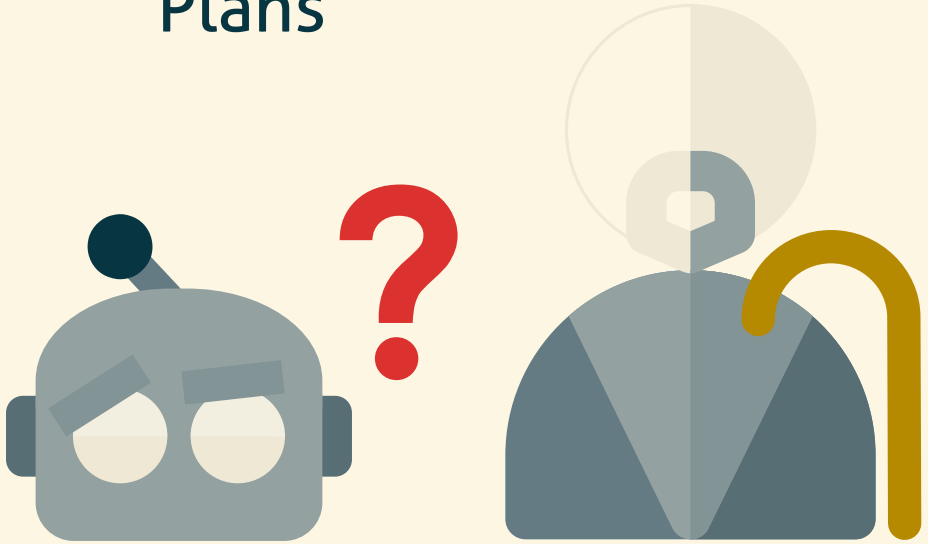


# 1.1 A what ?



- *Dependent people need help !*
  - Not **annoying** the person
  - Can't see *everything* they are doing
- How to help without asking ?
  - Guessing the intent somehow

- **Intent recognition**
  - Observed behavior → Goal
  - Using action sequences: Plans

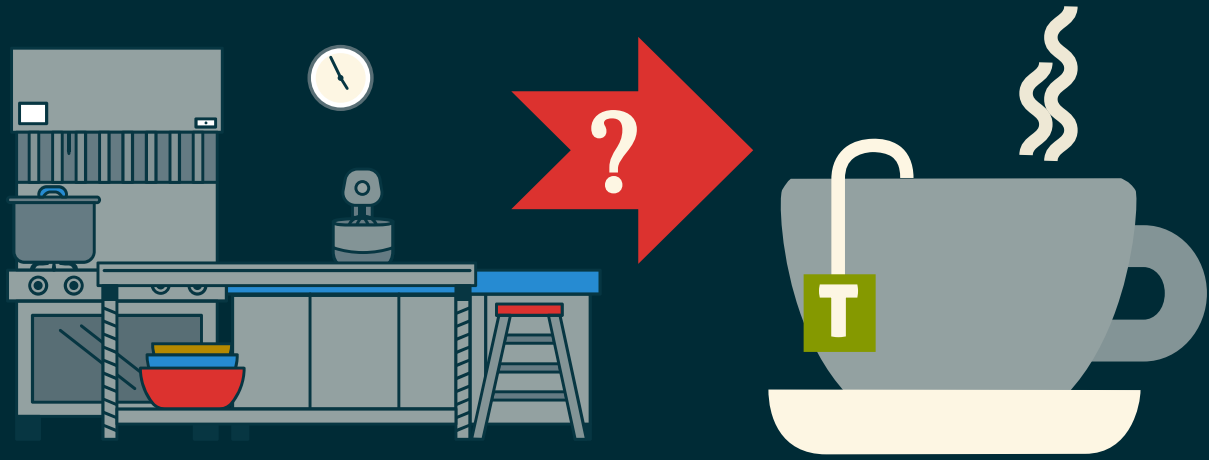


# 1.2 Kitchen Example



- Observation
  - Bob goes in the kitchen
- Possible goals
  - Bob cleans the dishes
  - Bob makes tea
  - …
- **Infer the correct goal**

- Issues
  - Multiple goals
  - Interleaving actions
  - Partial observations



# *Plan*

1 Introduction

2 Intent Recognition

3 Knowledge Representation

4 General Planning

5 Flexible Online Planning

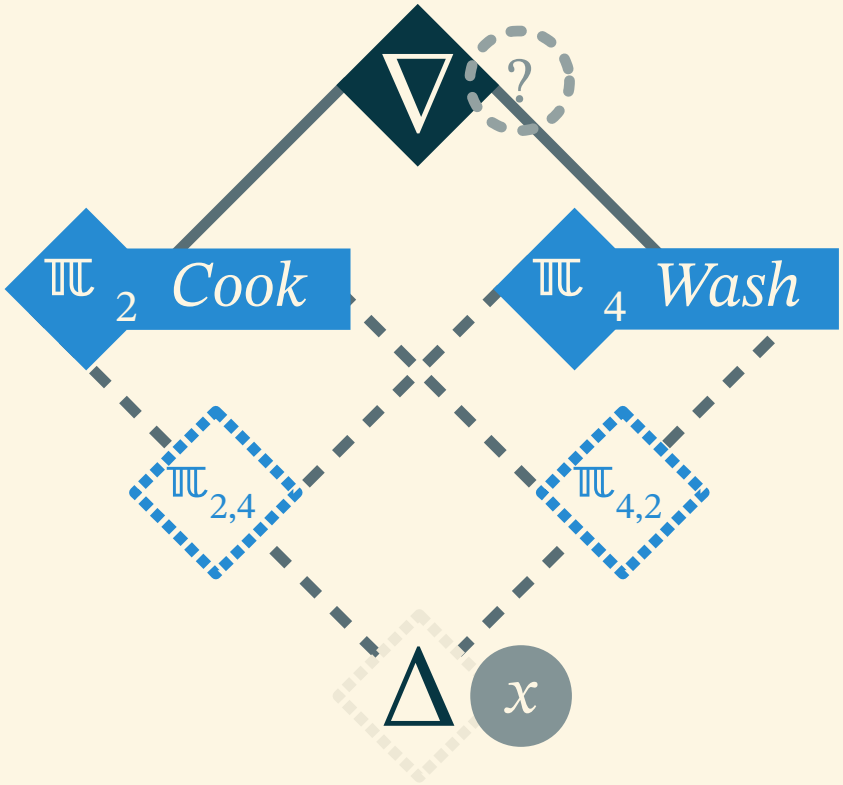
6 Conclusion

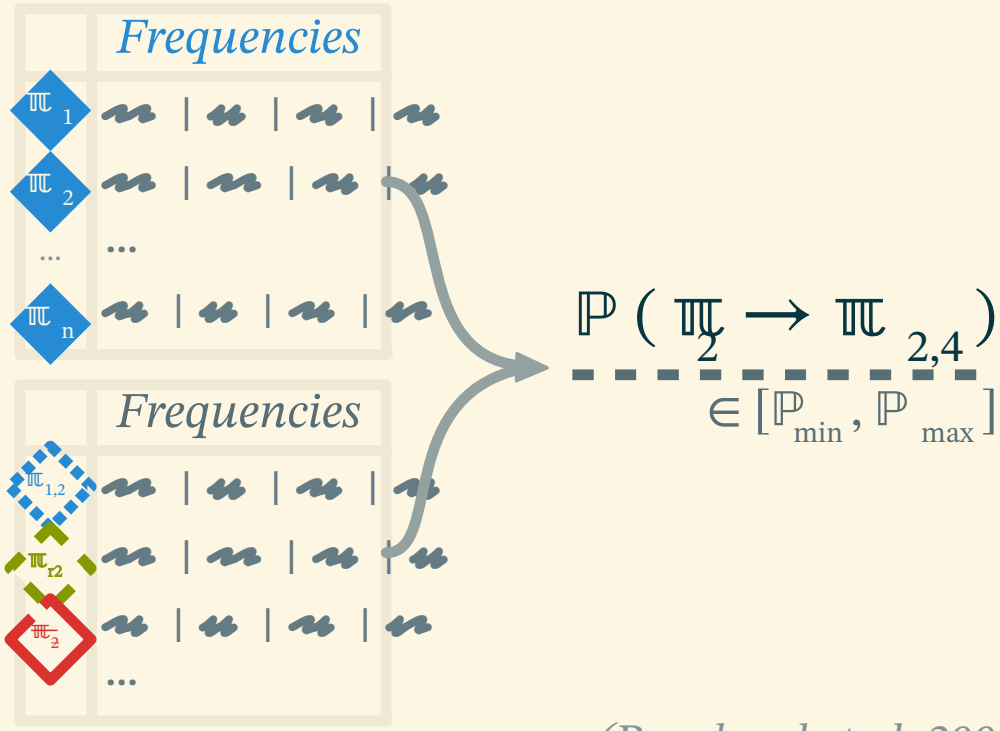
# 2 Intent Recognition

# 2.1 Logic Approach



- Lattice Based : ✔ Fast computations ✘ Exponential growth



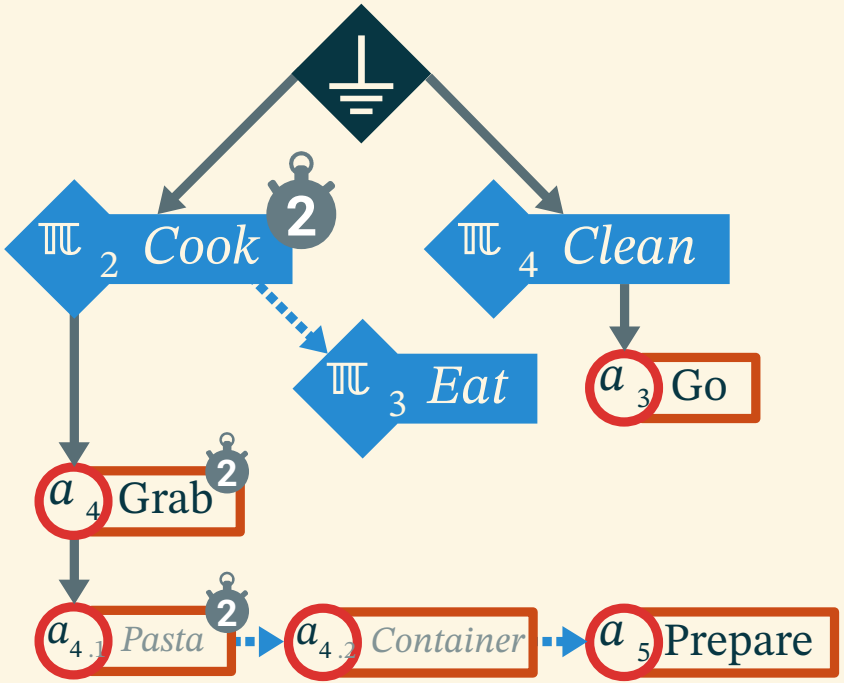



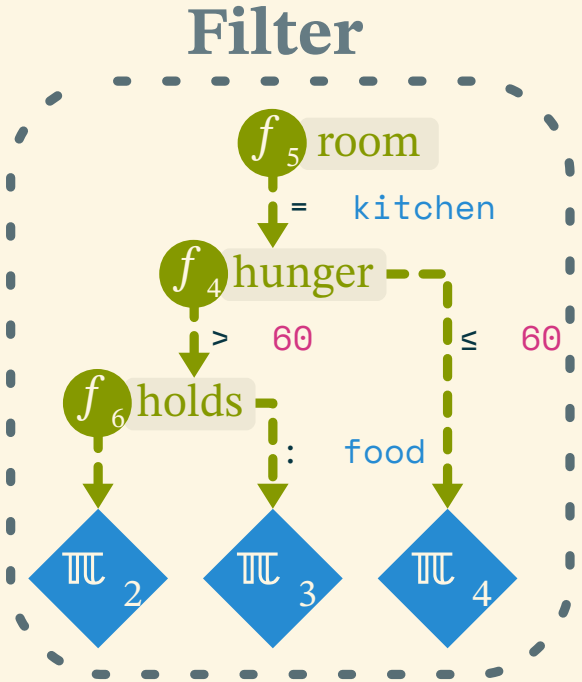

*(Bouchard et al. 2006)*

# 2.2 Stochastic Approach



- And/Or and decision tree
  - ✔ Accurate & efficient
  - ✘ Handmade plan library & tree



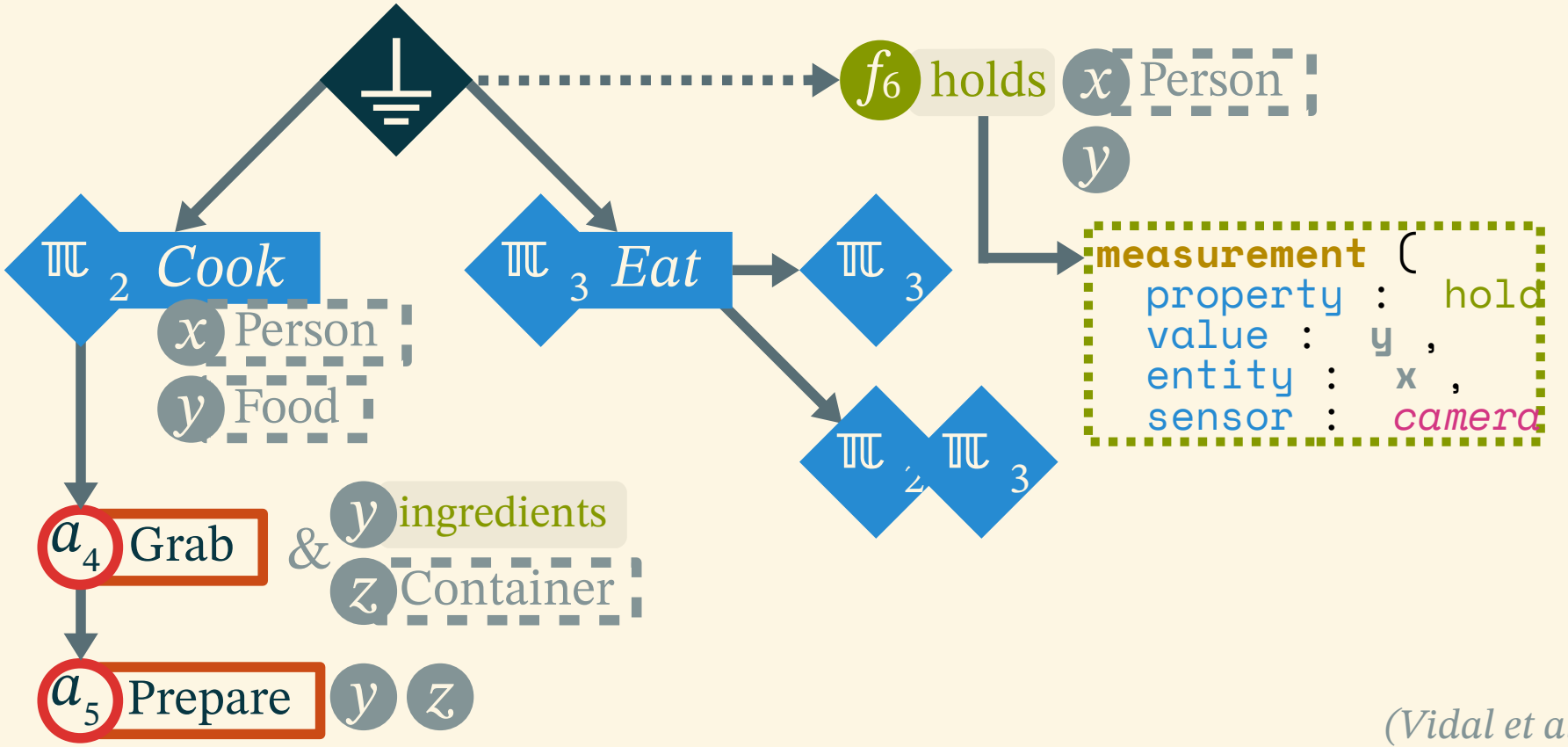




*(Avrahami et al. 2006)*

# 2.3 Grammatical Approach



- Valued Grammar : ✔ Versatile ✘ Slow refresh rate (~40s)

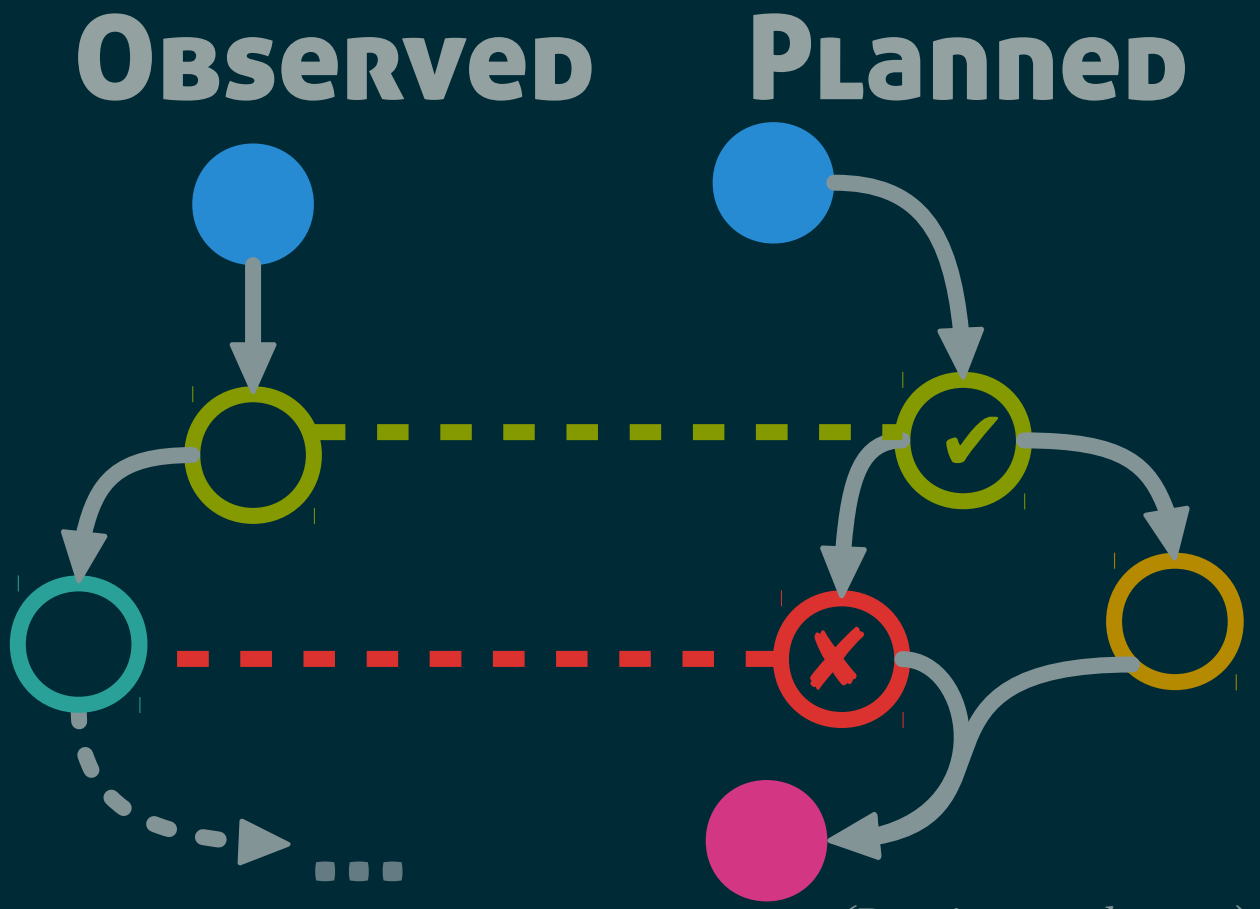



*(Vidal et al. 2010)*

# 2.4 Inverted Planning



- Intent Recognition
  - Find the goal of a plan
- **Planning**
  - Find the plan to a goal
- *Theory of Mind*
  - The easier the plan, the more likely the goal

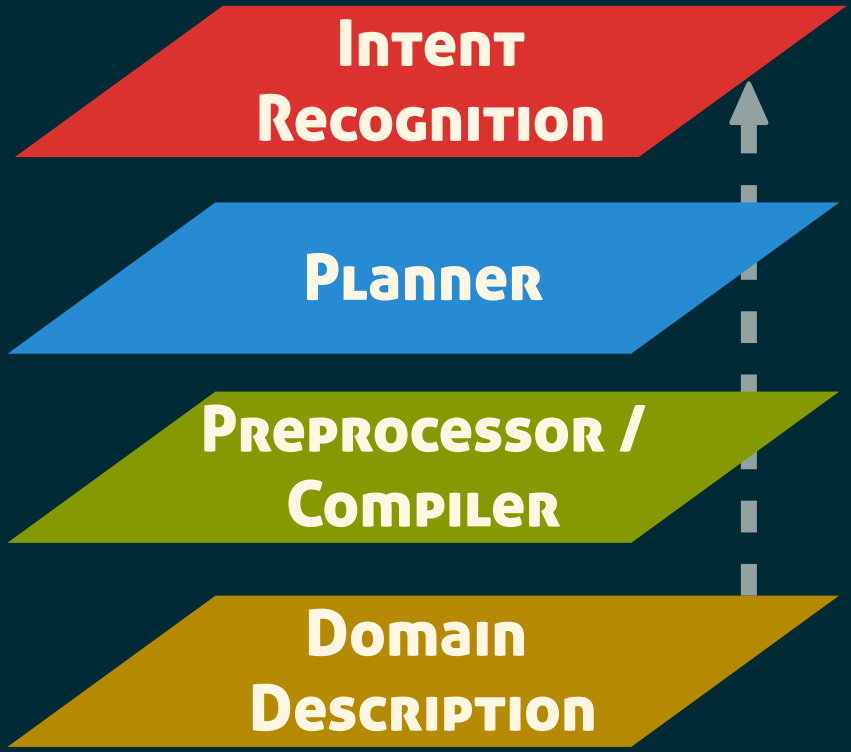



*(Ramirez et al. 2010)*

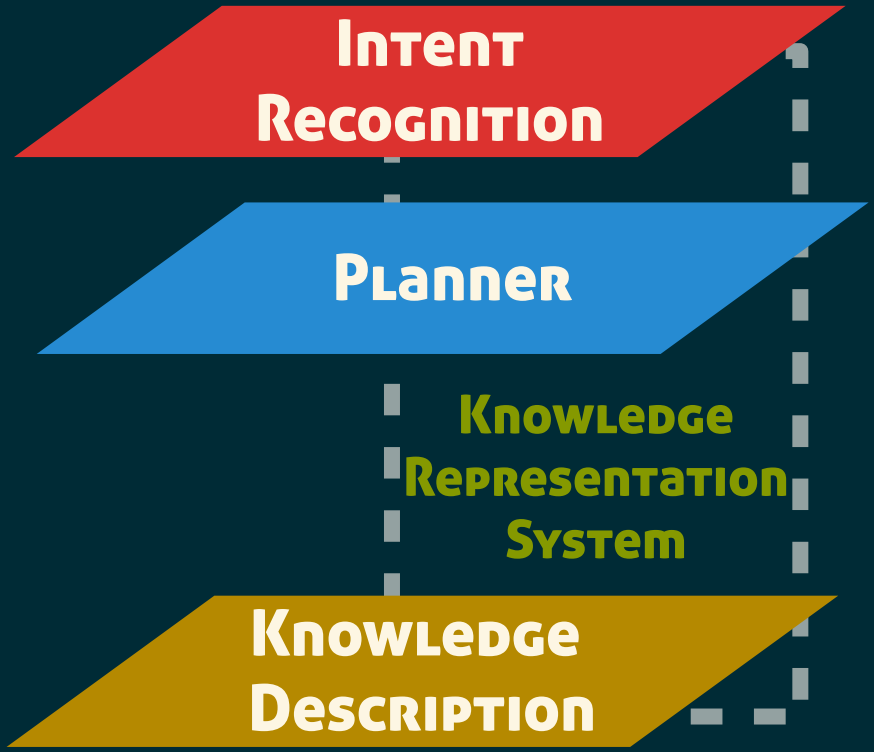
# 2.5 Framework Stacks



- Existing




- Contribution



# 3 Knowledge Representation

# 3.1 Knowledge in Planning

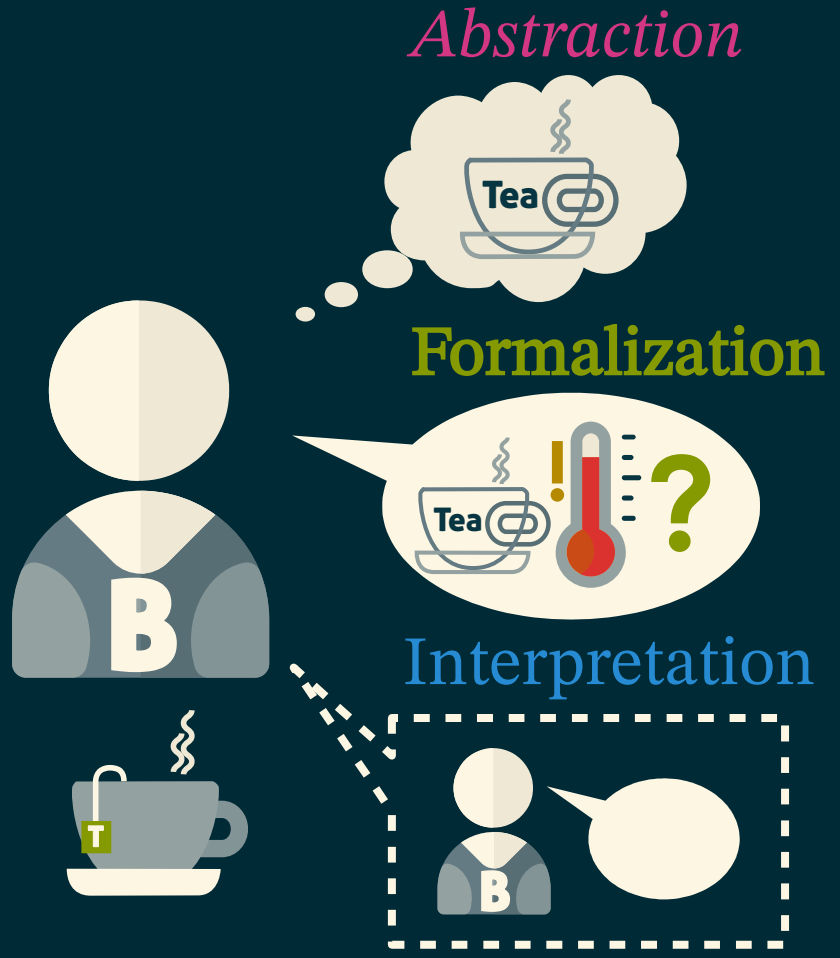


- **Reification** *(Cambridge Dictionary)*

> *"The act of changing something abstract into something real*

```
tea is not hot;

a
  eff    (tea is hot);
  method {(a1 → a2)};
```



# 3.2 Existing Tools



- Ontologies
  - Based on Description Logic

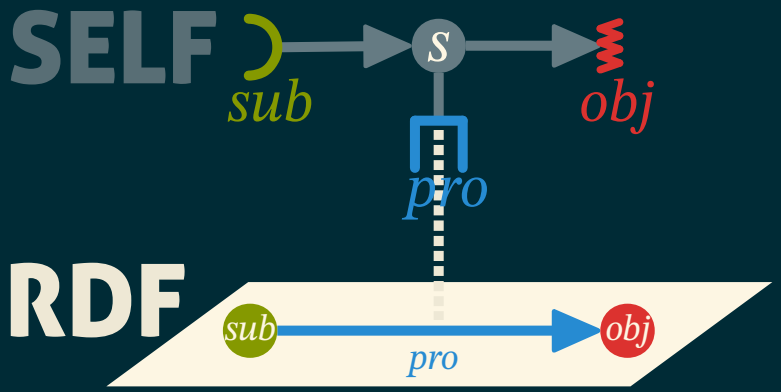
```
<?xml version="1.0"?>

<RDF>
  <Description about="Bob">
    <likes>Tea↙likes>
    <location>Kitchen↙location>
  ↙Description>
↙RDF>
```

- Languages
  - RDF
  - OWL-(Lite, DL, Full)
  - …
- Issues
  - Reification inefficient
  - Higher order knowledge
  - Flexibility of the structure

# 3.3 SELF Structurally Expressive Language Framework

- Minimal definition
  - Structure is meaning
  - **Ex:** `∀x = x;`
- More expressive
- Native reification
  - Express fluents and states in higher order spaces
  - Methods for hierarchical planning




Examples:

```
s = (bob @ kitchen);

a pre s;

a methods
{go(kitchen) → take(cup)};
```

*(Gréa et al. 2020)*

# 4 General Planning



# 4.1 Classical Planning



- Domain
  - Fluents
    - Formulas over objects
  - States
    - Properties of the world
    - Formulas over fluents
  - Actions
    - Preconditions
    - Effects
- Problem
  - Initial state
  - Goal state
- Plan (solution)
  - Action sequence
  - Order
    - Total
    - Partial

# 4.2 Example



- Fluents
  - thing *taken*
  - *hot* water, tea *ready*
- Actions
  - take, brew, boil, …

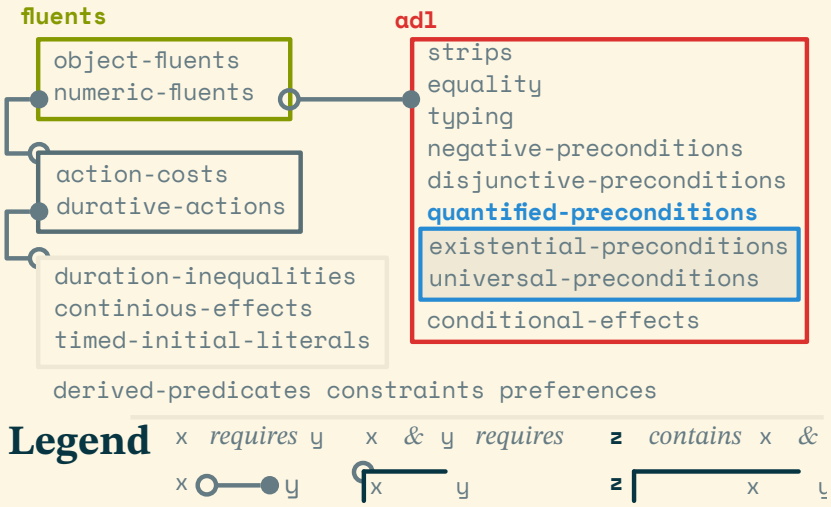

*Initial State*

**Goal State**

# 4.3 Existing Frameworks



- Standard language: **PDDL**
  - Numerous extensions to the language
  - Not used in probabilistic or hierarchical planning
  - Most of the time translated into an intermediate language for planners

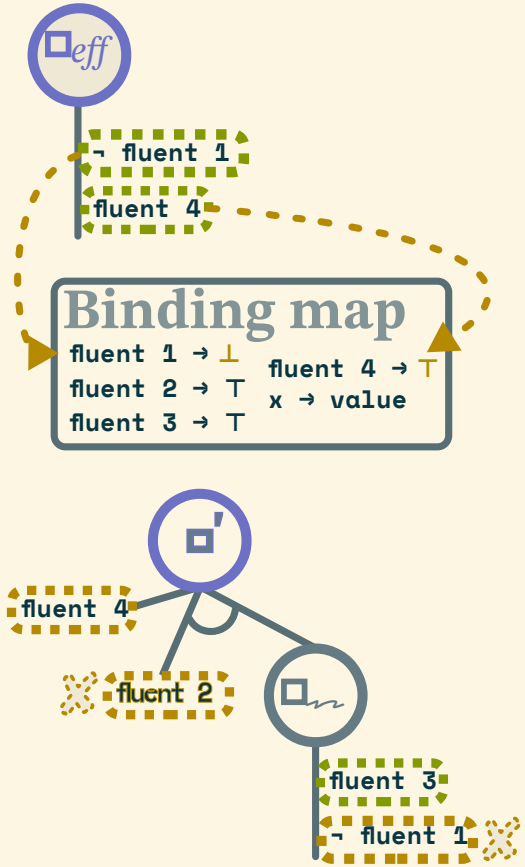


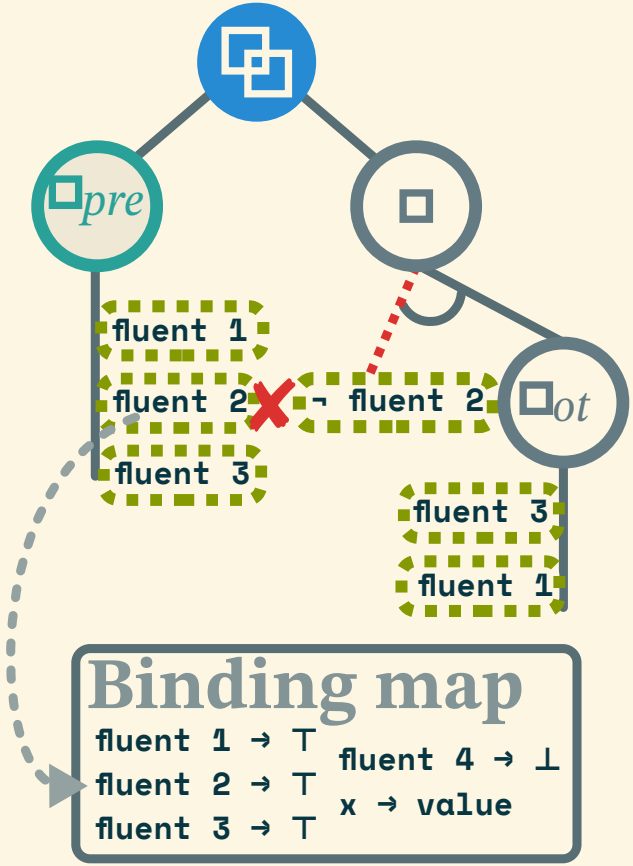

- Temporal
  - PDDL+
  - ANML
- Probabilistic
  - PPDDL
  - **RDDL**
- Multi-Agent
  - MAPL
  - MA-PDDL
- Hierarchical
  - UMCP
  - SHOP2
  - **HDDL**
  - HPDDL
- Ontological
  - *WebPDDL*
  - *OPT*
- Hybrids
  - SIADEX

# 4.4 Factorizing Planning States



## States: And/Or trees of Fluents




*(Gréa et al. 2020)*

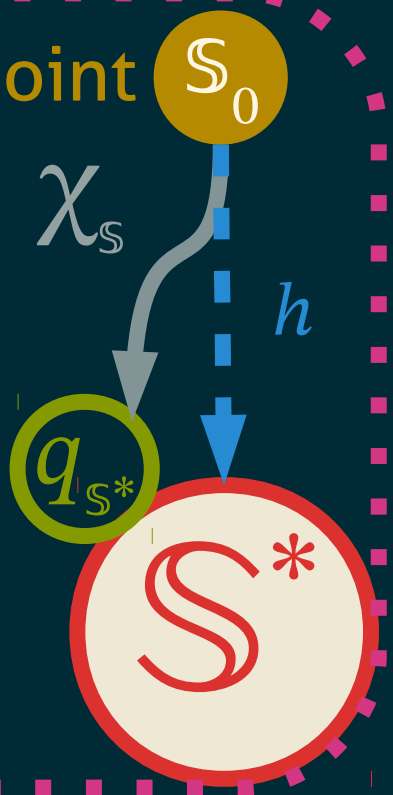
# 4.5 Planning Formalism Revisited



- Actions
  - Preconditions, Effects
  - Constraints
  - Cost, Duration, Probability
  - Methods
    - $(eff \rightarrow pre)$
- Problem
  - Root Action $\omega$
    - $pre(\omega) = a^0$
    - $eff(\omega) = a^*$

*(Gréa et al. 2020)*

- Search Space
  - Starting point
  - Iterator
  - Heuristic
  - Solution predicate
  - Solutions




- Instances for
  - **State-transition**
  - Plan space
  - Case based
  - Probabilistic
  - Hierarchical

$\mathbb{S}_0 = eff(\omega)$ (initial)
$\chi_{\mathbb{S}} = a \in A$ (actions)
$q_{\mathbb{S}^*} = (\vDash pre(\omega))$ (goal)
$\mathbb{S} = \square$ (states)

*(Gréa et al. 2020)*

# 4.7 PDDL vs COLOR



```
(define (domain tea)
  (:requirements :equality :object-fluents)
  (:types container, liquid, item)
  (:constants no-item - item, water - liquid, cup
- container)
  (:predicates (hot ?x - liquid))
  (:functions (taken) - item)

  (:action take
      :parameters (?x - item)
      :precondition (and (= (taken ?x) no-item))
      :effect (and (assign (taken) ?x)))
  (:action heat
      :parameters (?x - liquid)
      :precondition (and (not (hot ?x))
                  (= (taken ?x) ?x))
      :effect (and (hot ?x))
```

```
"planning.w" = ? ;
take(item) pre (taken(~), ?(item));
take(item) eff (taken(item));

heat(thing) pre (~(hot(thing)), taken(thing));
heat(thing) eff (hot(thing));

make(drink) method (
    init(make(drink)) → take(spoon),
    take(spoon) → put(spoon),
    init(make(drink)) → infuse(drink,water,cup),
    infuse(drink,water,cup) → take(cup),
    take(cup) → put(cup),
    put(spoon) → goal(make(drink)),
    infuse(drink,water,cup) → goal(make(drink)),
    put(cup) → goal(make(drink))
);
```

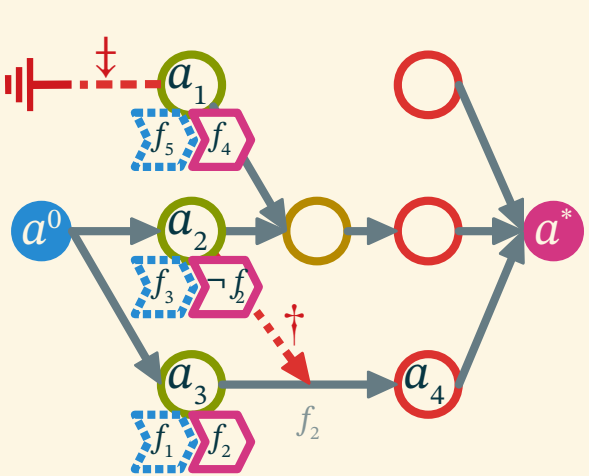
*(Gréa et al. 2020)*

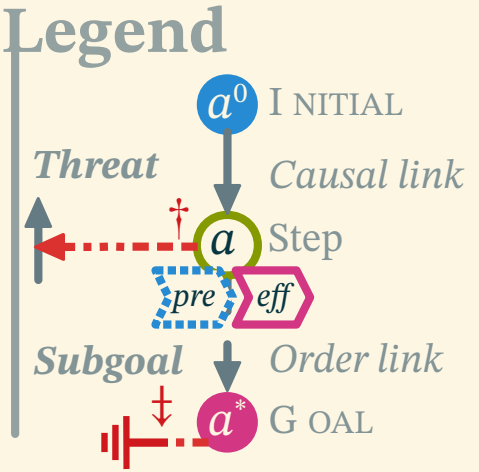
# 5 Flexible Online Planning

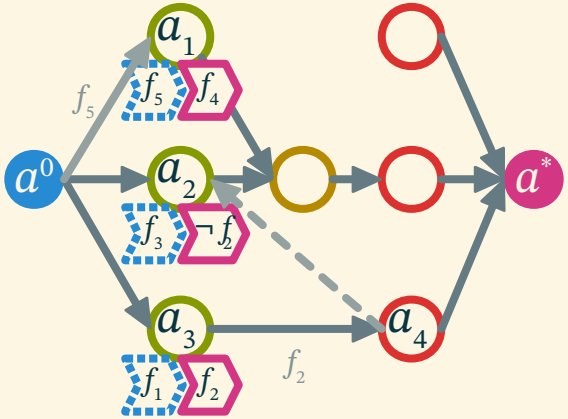
# 5.1 Plan Space Planning

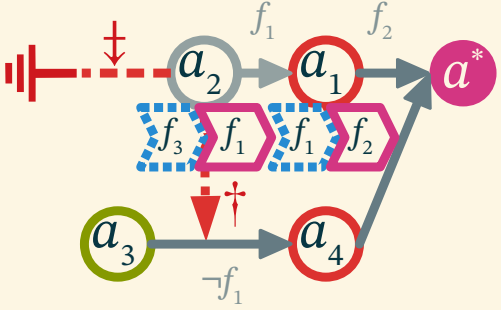


- Exploration by refinements
- Flaws
  - Subgoals
  - Threats
- Resolvers
  - Side effects
- May need backtracking



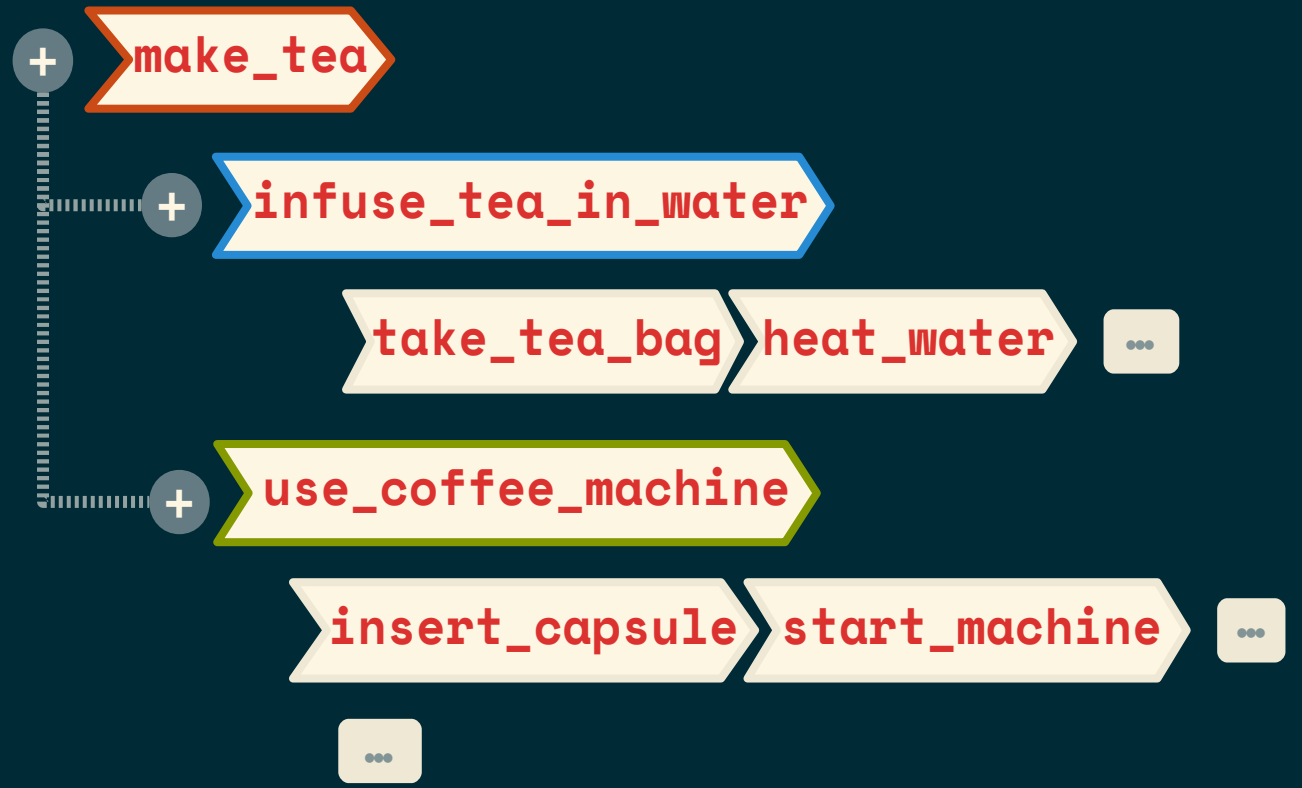



# 5.2 Hierarchical Task Networks



- Based on tasks decomposition
  - Replace task with method
- Numerous approaches



# 5.3 Planning Phases

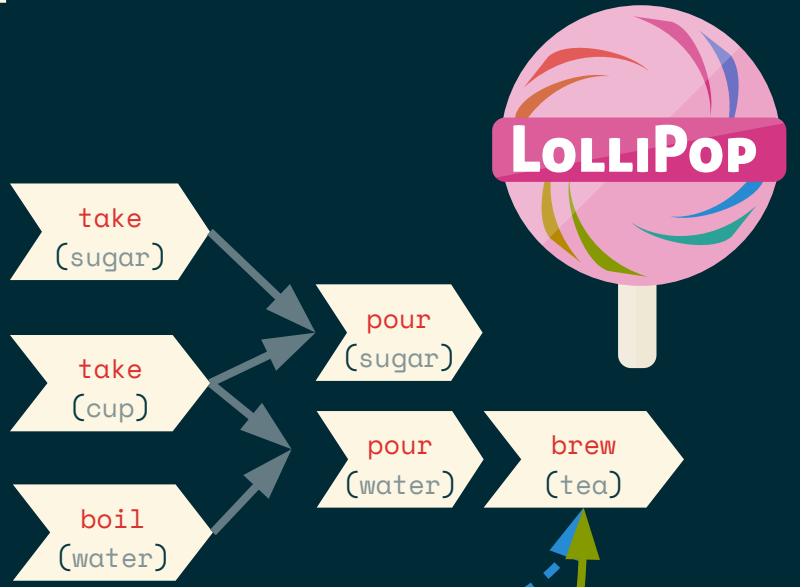


- Phases dependent on
  - Available information
  - Timing constraints
  - Planning paradigm

# 5.4 Plan Repair Prototype



- Partial Order Planner (POP)
- Operator dependency graph
- Negative refinements
- Alternatives & Orphans





*Orphan* *Alternative*


**Subgoal**

- Utility Heuristics

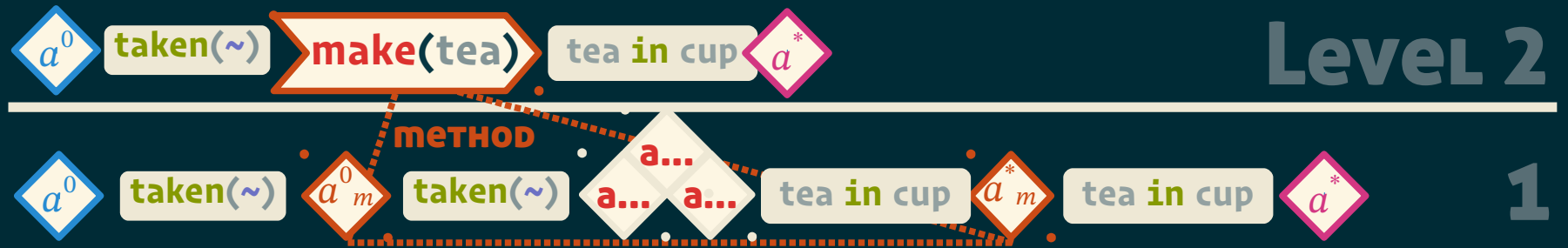
*(Gréa et al. 2016)*

# 5.5 Abstract Planning



- HTN + POP planning
- Partial Resolution
  - An abstract solution at every level of abstraction
- Search by level
  - Expansion after completion :

- Decomposition flaw
  - Resolver : Decompose one composite action in the plan

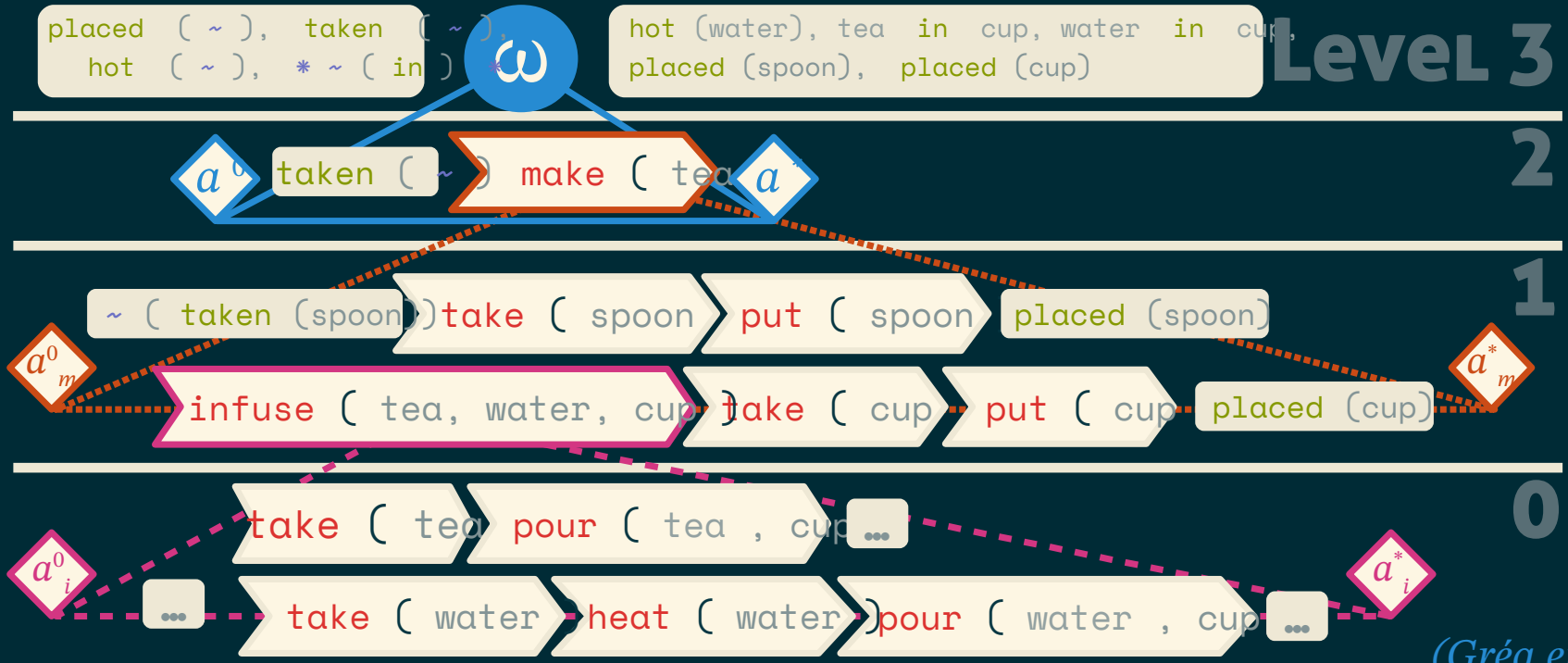



(Gréa et al. 2019)

# 5.6 HEART



- Low priority for expansion
- Each level is a plan (abstract solution)
- Change of level
  - Propagation of atomic actions
  - Expansion of Composite Equities

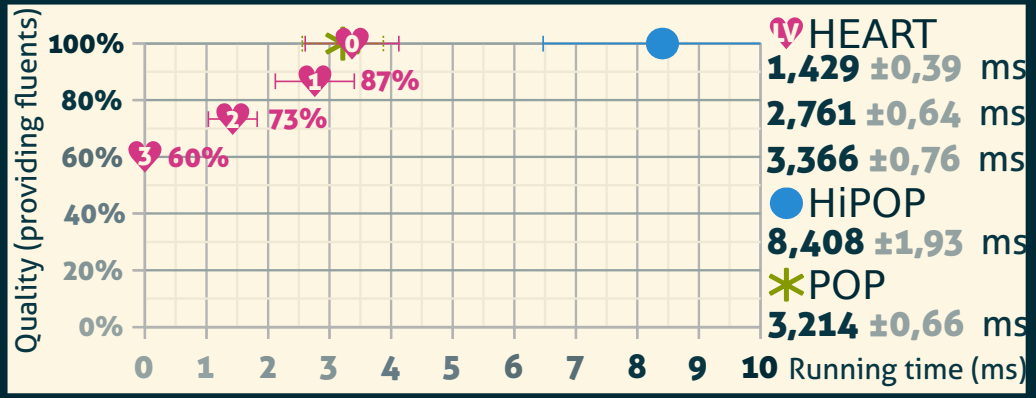


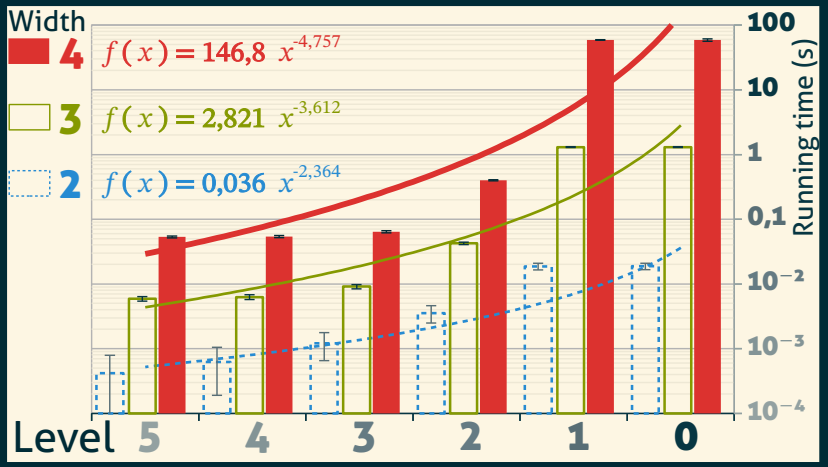

(Gréa et al. 2019)

# 5.7 Results



- 60% of the fluents **before** planning
- Exponentially faster at high abstraction levels
- Faster than HiPOP on some problems
- Common problems solved in milliseconds!

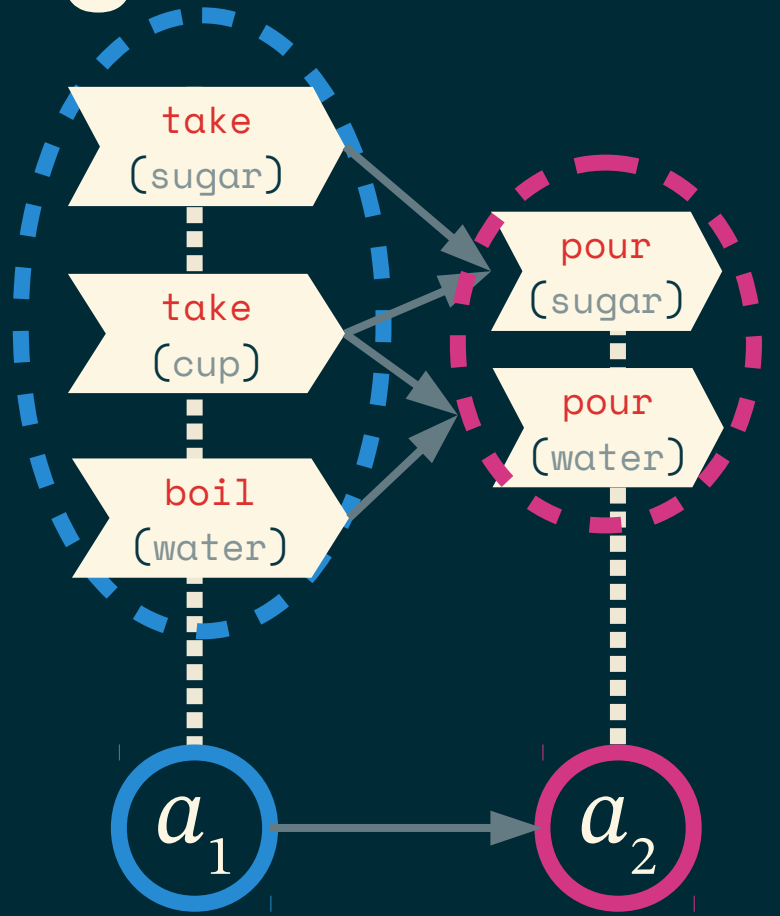



*(Gréa et al. 2019)*

# 5.8 Toward Intent Recognition

- Linearized parallel actions using graph quotient
- Abstraction makes it easier (smaller plans)
- Backward chaining is inefficient




*(Gréa et al. 2020)*

# 6 Conclusion



# 6.1 Contributions & Results



- SELF: A knowledge description language defined by structure
- COLOR: A general framework for planning with its formalization
- LOLLIPOP: A plan repair planner for online planning
- HEART: A flexible approach to real-time planning for abstract planning

*(Gréa et al. 2020)*

# 6.2 Perspectives



- SELF Improvement
  - Improve the instantiation workflow
  - Parameterize flexibility performances
- Planning Colorized
  - Conversion tool from PDDL
- Fixing Planning Domains
  - Allow HEART to discover new HTN methods (macro-action learning)

# Thanks for listening !