Endomorphic metalanguage and abstract planning for real-time intent recognition

Antoine Gréa

Lyon 1

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

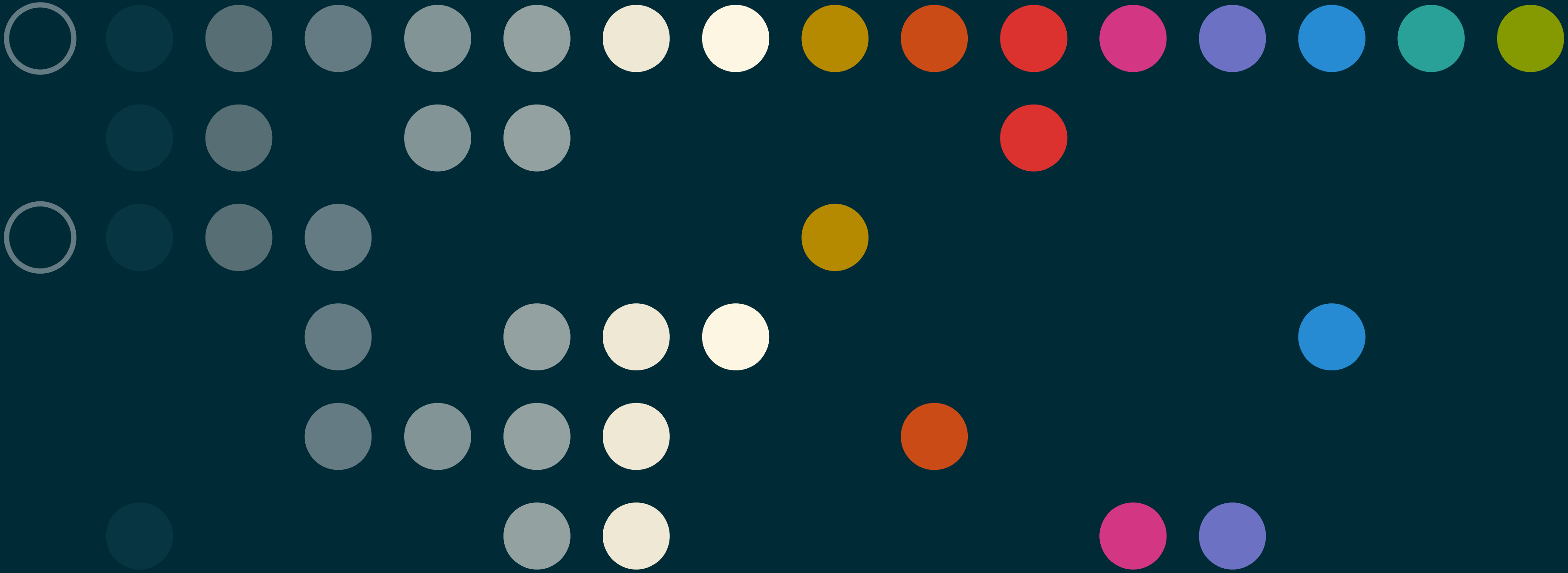
- Directors
  - Samir Aknine
  - Lætitia Matignon
- Jury
  - Hamamache Kheddouci
  - Ivan Varzinczac
- Reviewers
  - Eva Onaindia
  - Damien Pellier

Endomorphic metalanguage and abstract planning for real-time intent recognition

LIRIS Lyon 1

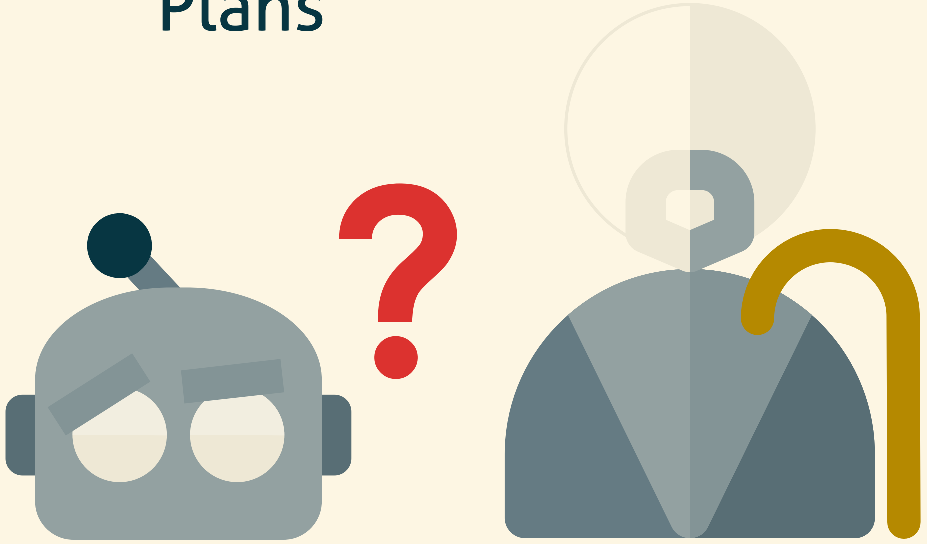
Antoine Gréa

1 Introduction



A what ?

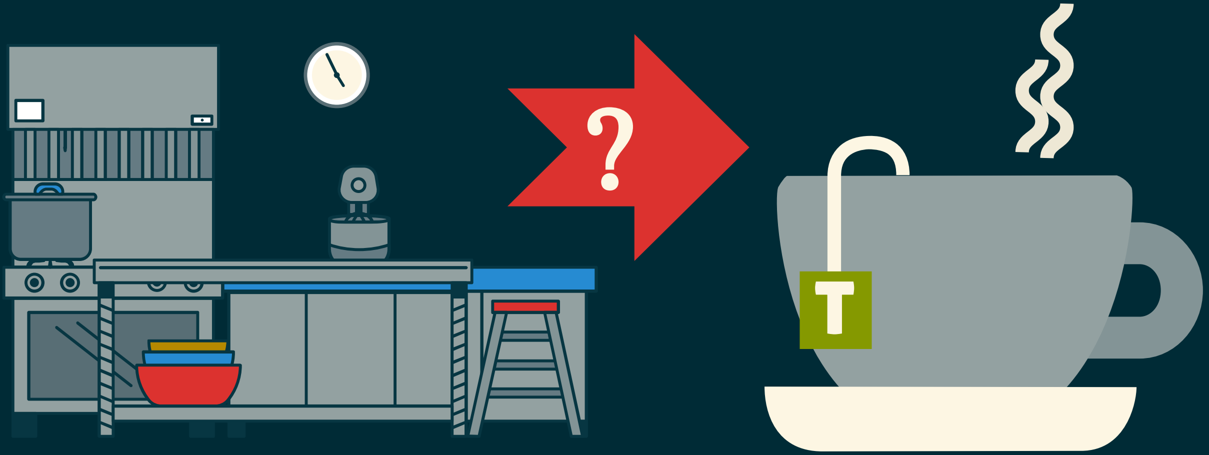
- *Dependent people need help !*
  - Not **annoying** the person
  - Can't see *everything* they are doing
- How to help without asking ?
  - Guessing the intent somehow
- **Intent recognition**
  - Observed behavior → Goal
  - Using action sequences: Plans



Kitchen Example

- Observation
  - Bob goes in the kitchen
- Available goals
  - Bob cleans the dishes
  - Bob makes tea
  - …
- Infer correct one

- Issues
  - Multiple goals
  - Interleaving
  - Partial Observation



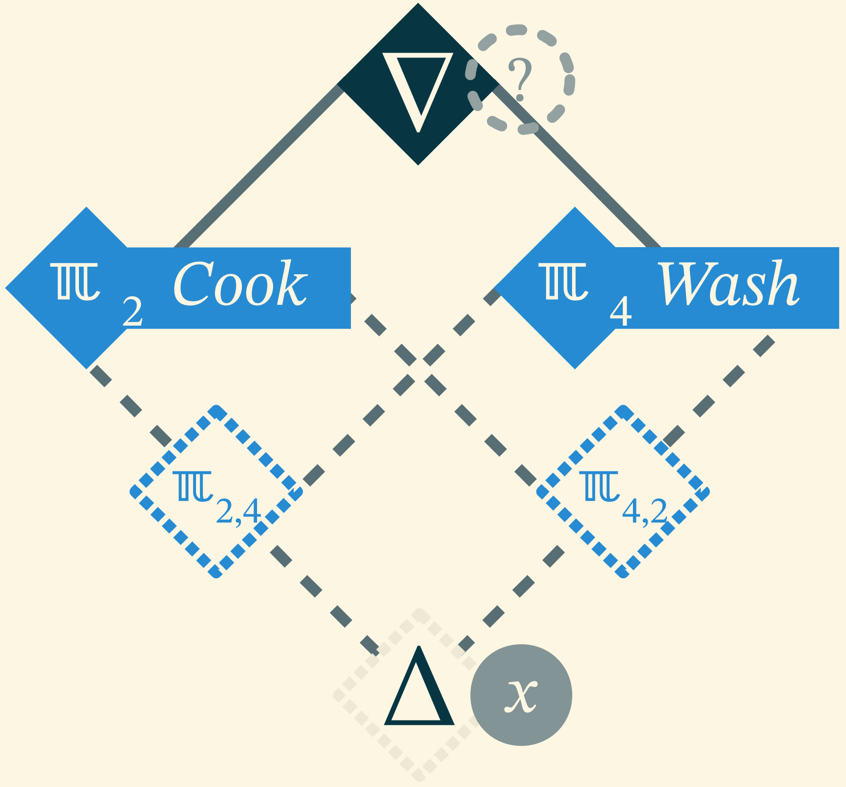
Plan

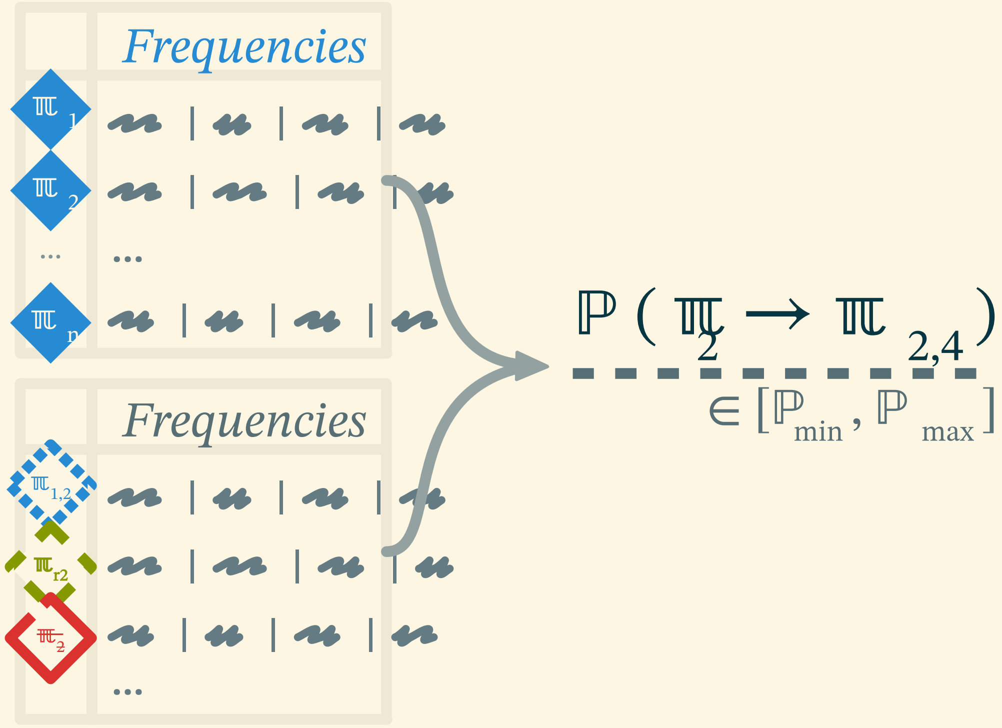
2 Intent Recognition

2.1 Logic Approach

[@bouchard_2006]

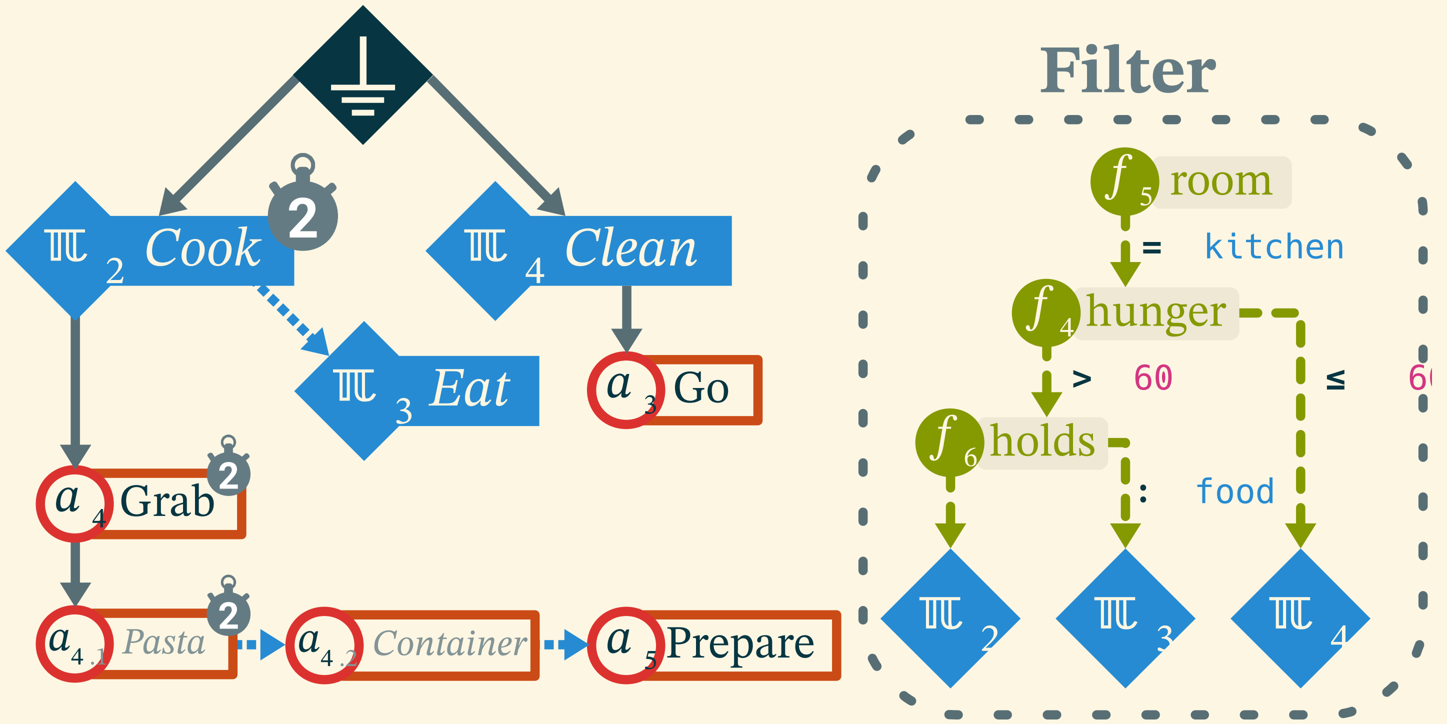
- Lattice Based : ✓ Fast computations ✗ Exponential growth




Frequencies
Frequencies
$\mathbb{P}(\pi_2 \rightarrow \pi_{2,4})$
$\in [\mathbb{P}_{min}, \mathbb{P}_{max}]$

2.2 Stochastic Approach

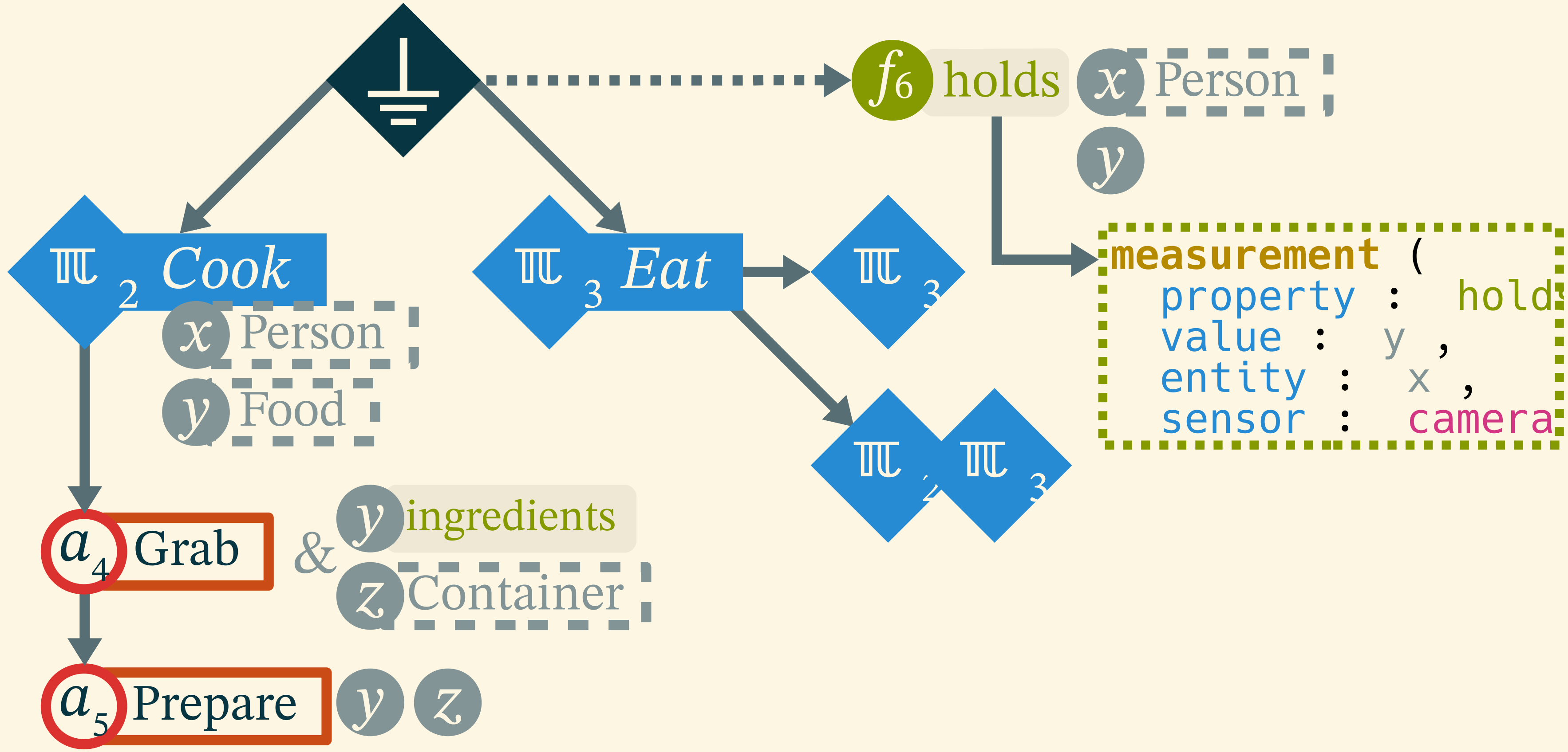
- And/Or and decision tree :
  - ✓ Accurate & efficient
  - ✘ Handmade plan library & tree



2.3 Grammatical Approach

[@vidal_2010]

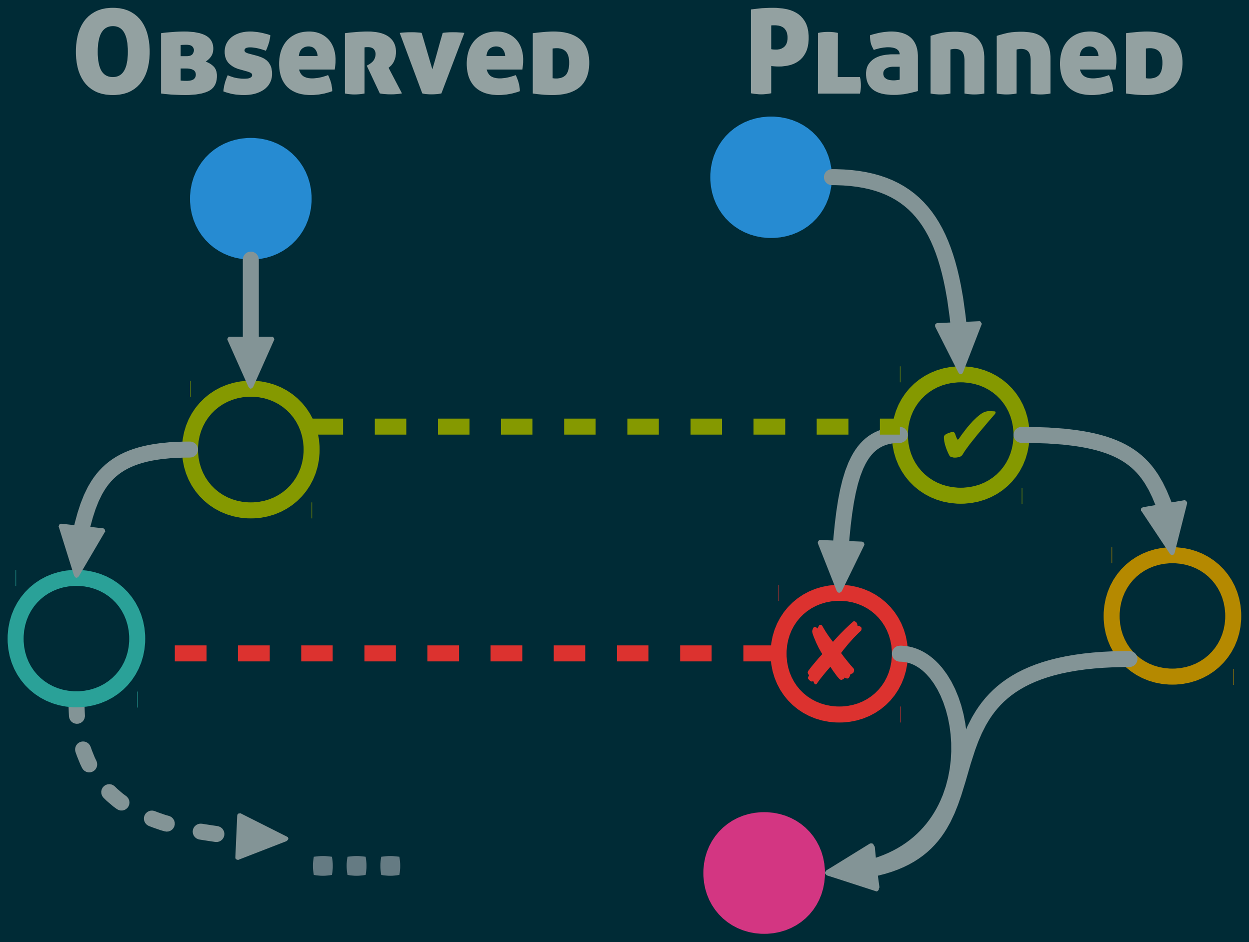
- Valued Grammar : ✓ Versatile ✗ Slow refresh rate (~40s)



```
measurement (
  property : holds
  value : y ,
  entity : x ,
  sensor : camera
```

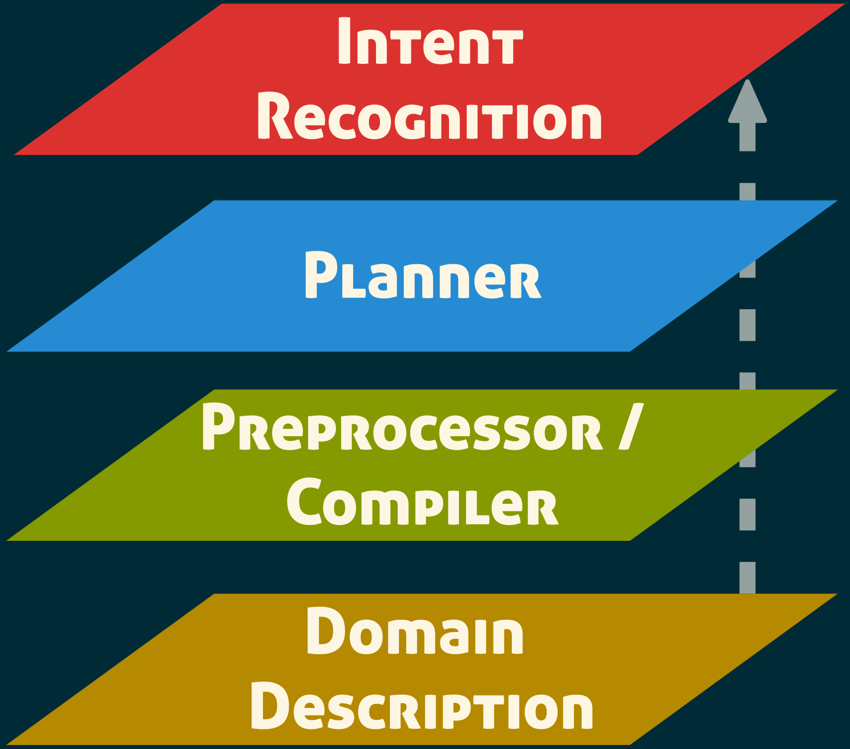
2.4 Inverted Planning

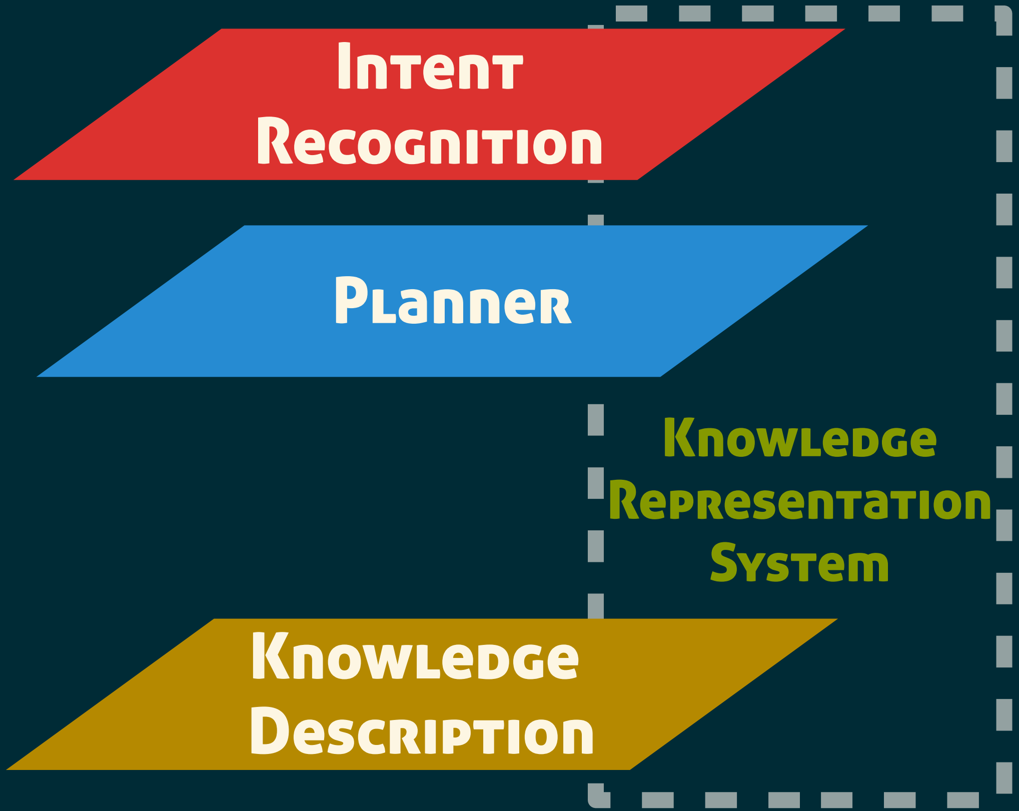
- Intent Recognition
  - Find the goal of a plan
- **Planning**
  - Find the plan to a goal
- *Theory of Mind*
  - The easier the plan, the more likely the goal



2.5 Framework Stacks

- Existing
- Contribution





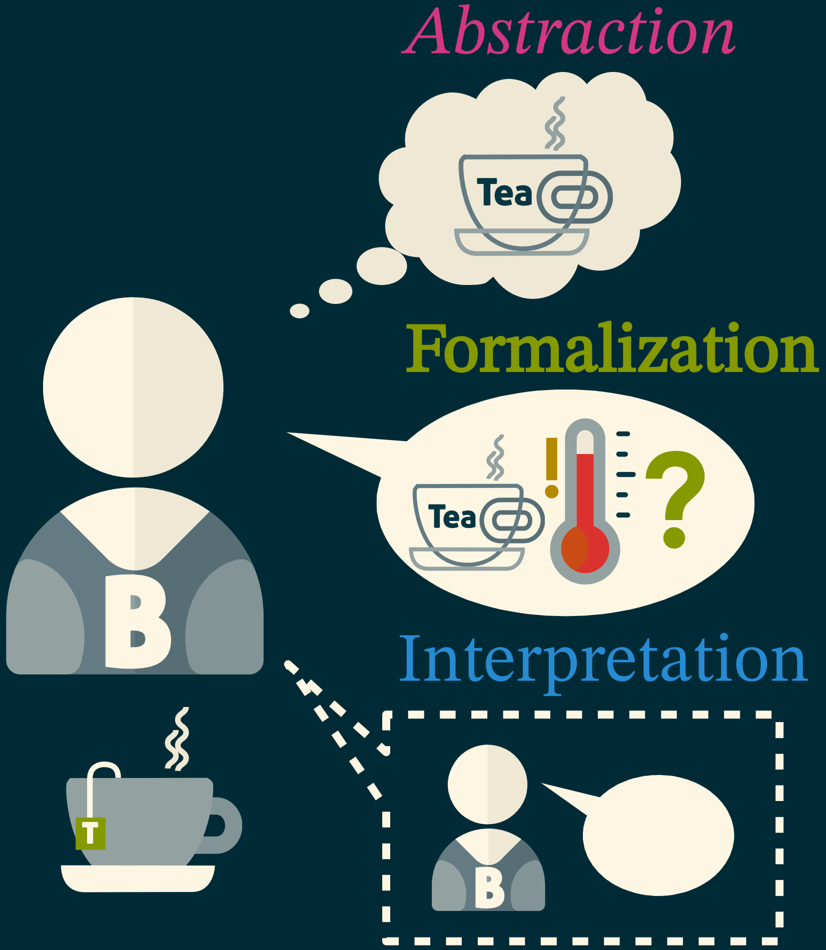
3 Knowledge Representation

Knowledge in Planning

- **Reification** [@cambridge]

“*The act of changing something abstract into something real*

```
tea is not hot;

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



Existing Tools

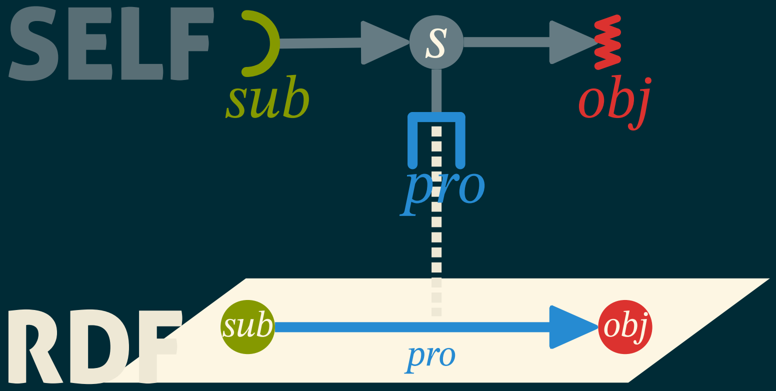
- Ontologies
  - Based on Description Logic

```
<?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

SELF

- Self defined
  - Structure = 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)};
```

4 General Planning

Classical Planning

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

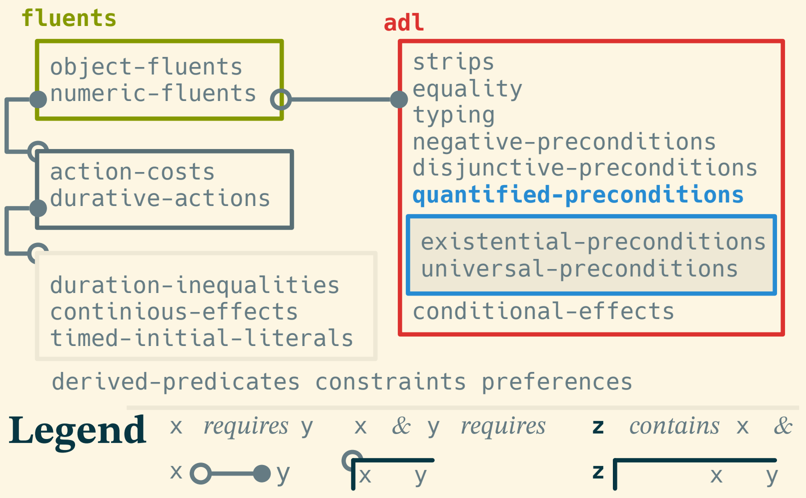
Example

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

Sugar
Sugar
T
T
Initial State
Goal State

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

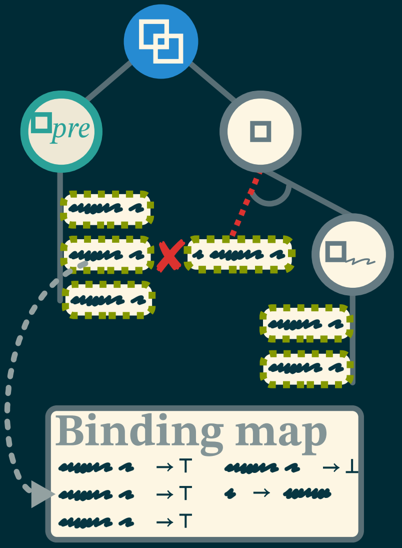


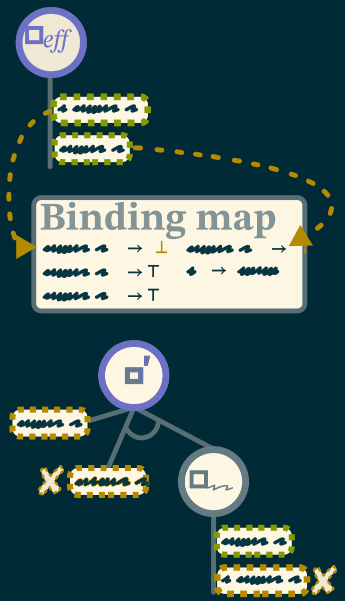
- Temporal
  - PDDL+
  - ANML
- Probabilistic
  - PPDDL
  - **RDDL**
- Multi-Agent
  - MAPL
  - MA-PDDL
- Hierarchical
  - UMCP
  - SHOP2
  - **HDDL**
  - HPDDL
- Ontological
  - *WebPDDL*
  - *OPT*
- Hybrids
  - SIADEX

Planning Formalism Revisited [@Gréa]

- States
  - And/Or trees of Fluents

Verifying Applying

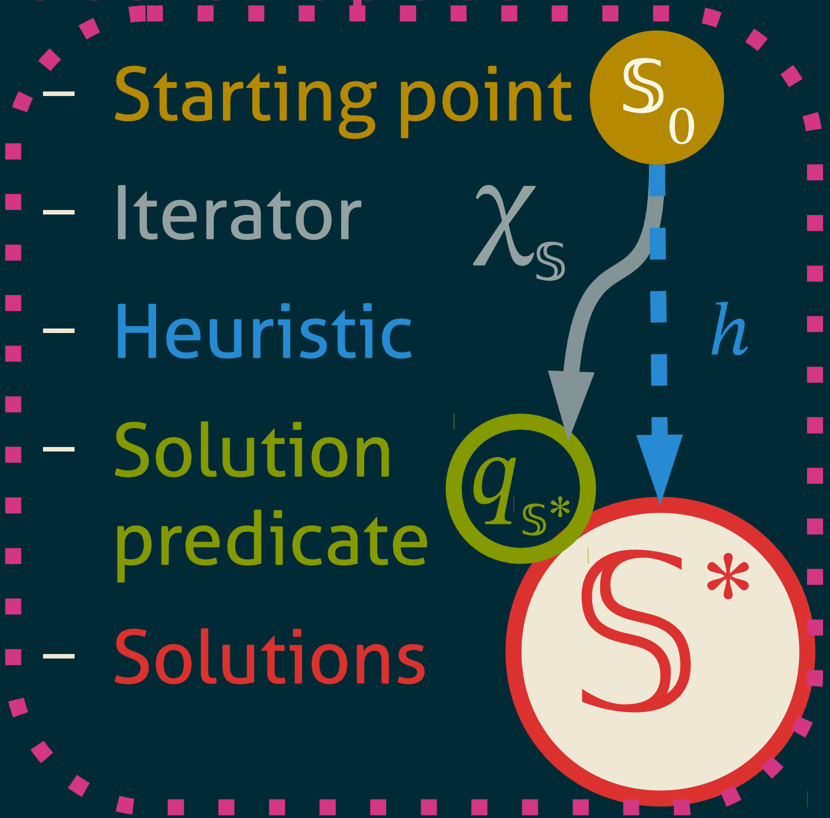




- Actions
  - Precondition, Effects
  - Constraints
  - Cost, Duration, Probability
  - Methods (`eff → pre`)

General Planning Framework [@Gréa]

- Search Space
  - Starting point $\mathbb{S}_0$
  - Iterator $\chi_{\mathbb{S}}$
  - Heuristic h
  - Solution predicate $q_{\mathbb{S}^*}$
  - Solutions $\mathbb{S}^*$



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

$\mathbb{S}_0 = a^0$ (initial)

$\chi_{\mathbb{S}} = a \in A$ (actions)

$q_{\mathbb{S}^*} = (=a^*)$ (goal)

$\mathbb{S} = \square$ (states)

COLOR Framework

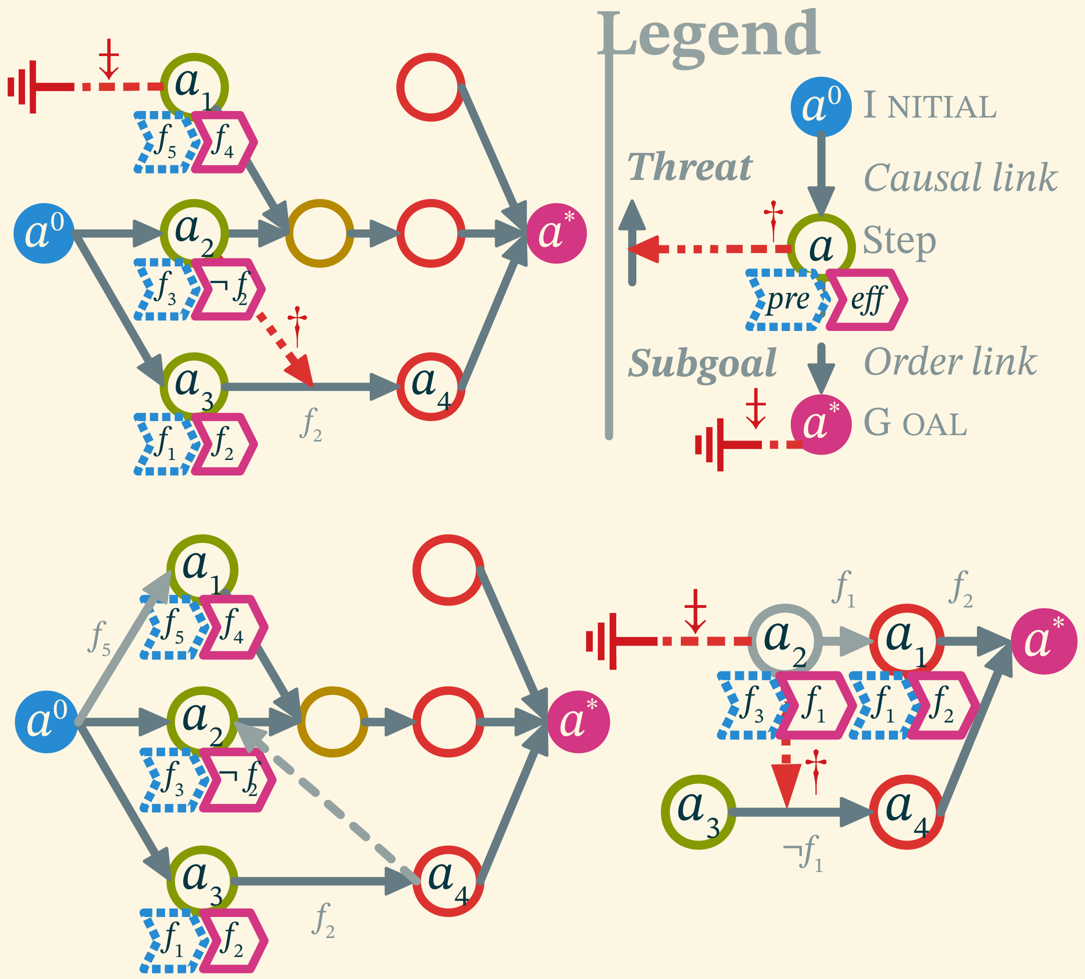
```
(define (domain tea)
  (:requirements :equality :object-fluents)
  (:types container, liquid, item)
  (:constants no-item – item, water – liquid, cup –
container)
  (:predicates (hot ?x - liquid))
  (:functions (taken) - item)

  (: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))
);
```

5 Flexible Online Planning

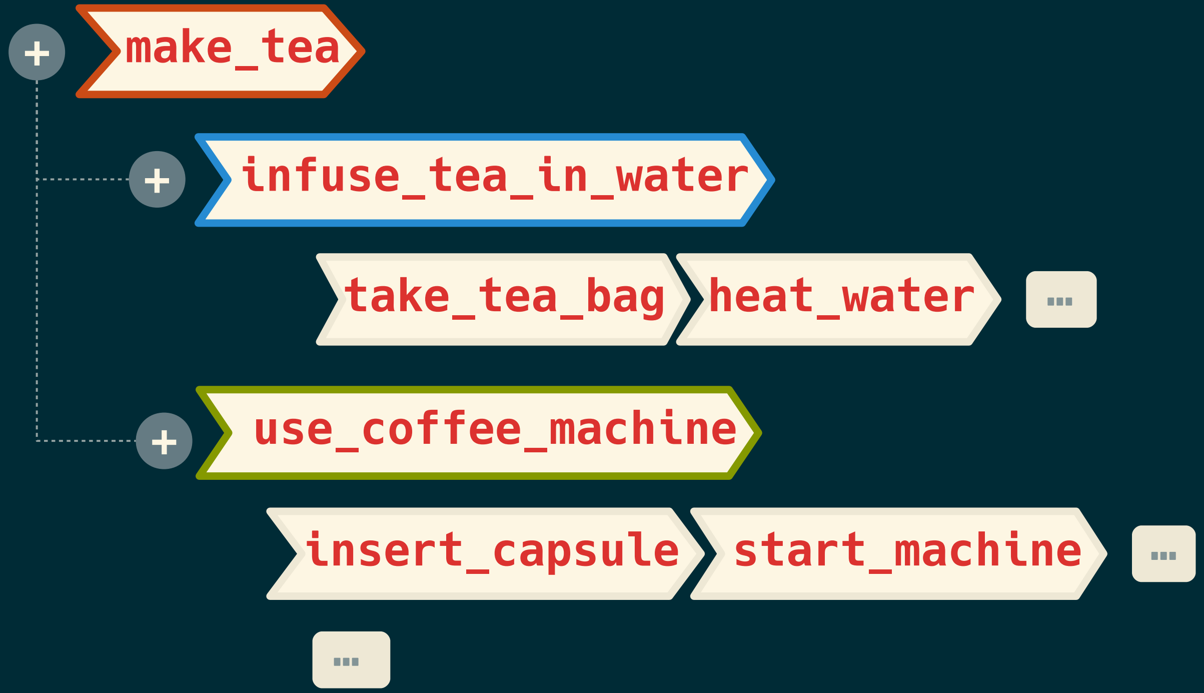
Plan Space Planning

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



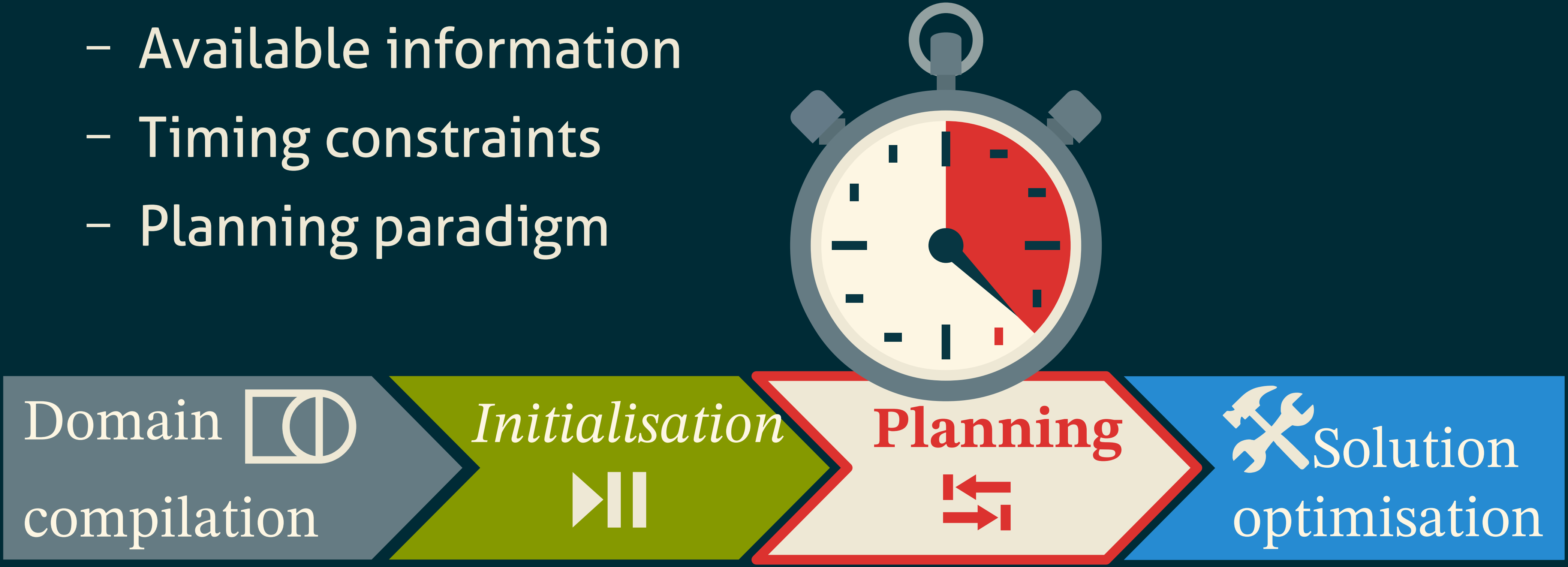
Hierarchical Task Networks

- Based on tasks decomposition
  - Replace task with method
- Lots of different approaches



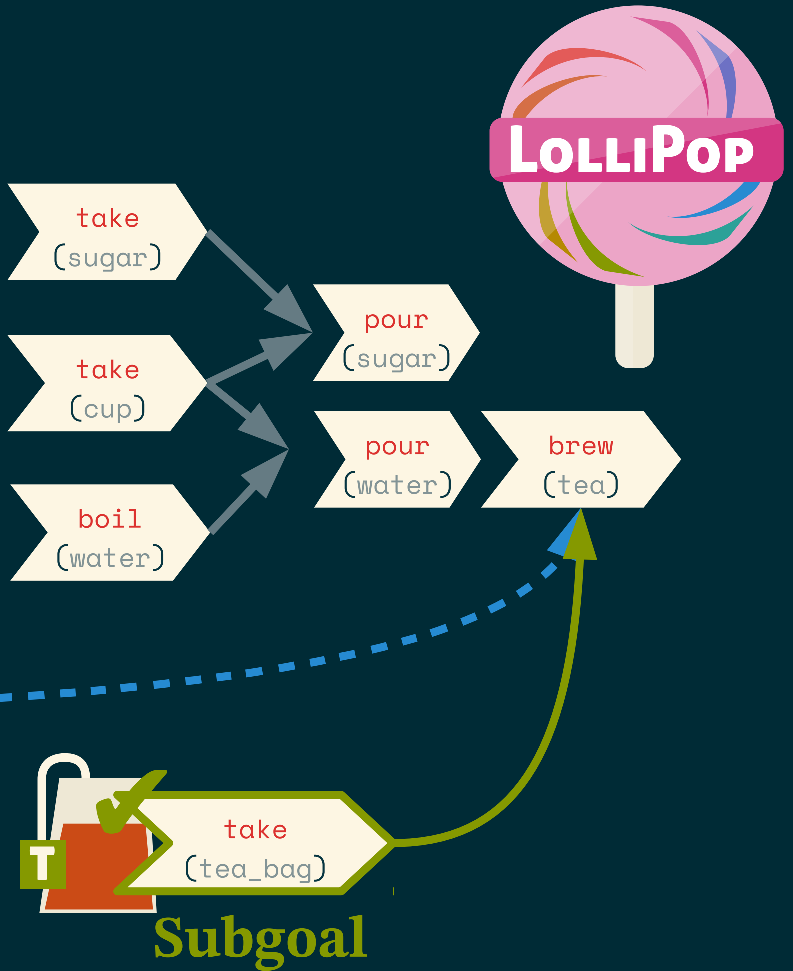
Planning Phases

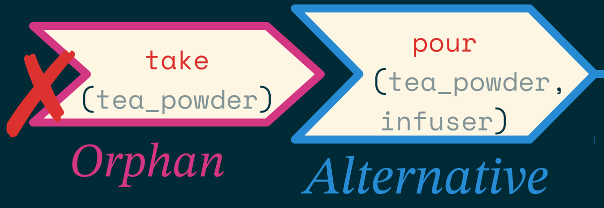
- Phases dependent on
  - Available information
  - Timing constraints
  - Planning paradigm



Plan Repair Prototype

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



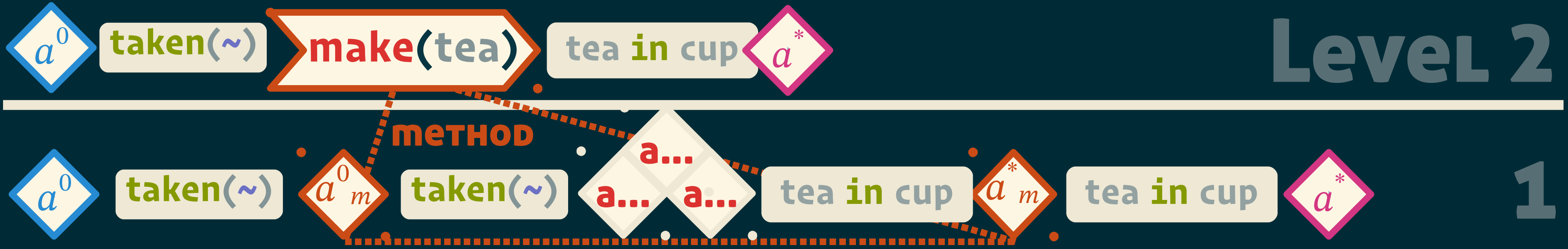


- Utility Heuristics

Abstract Planning

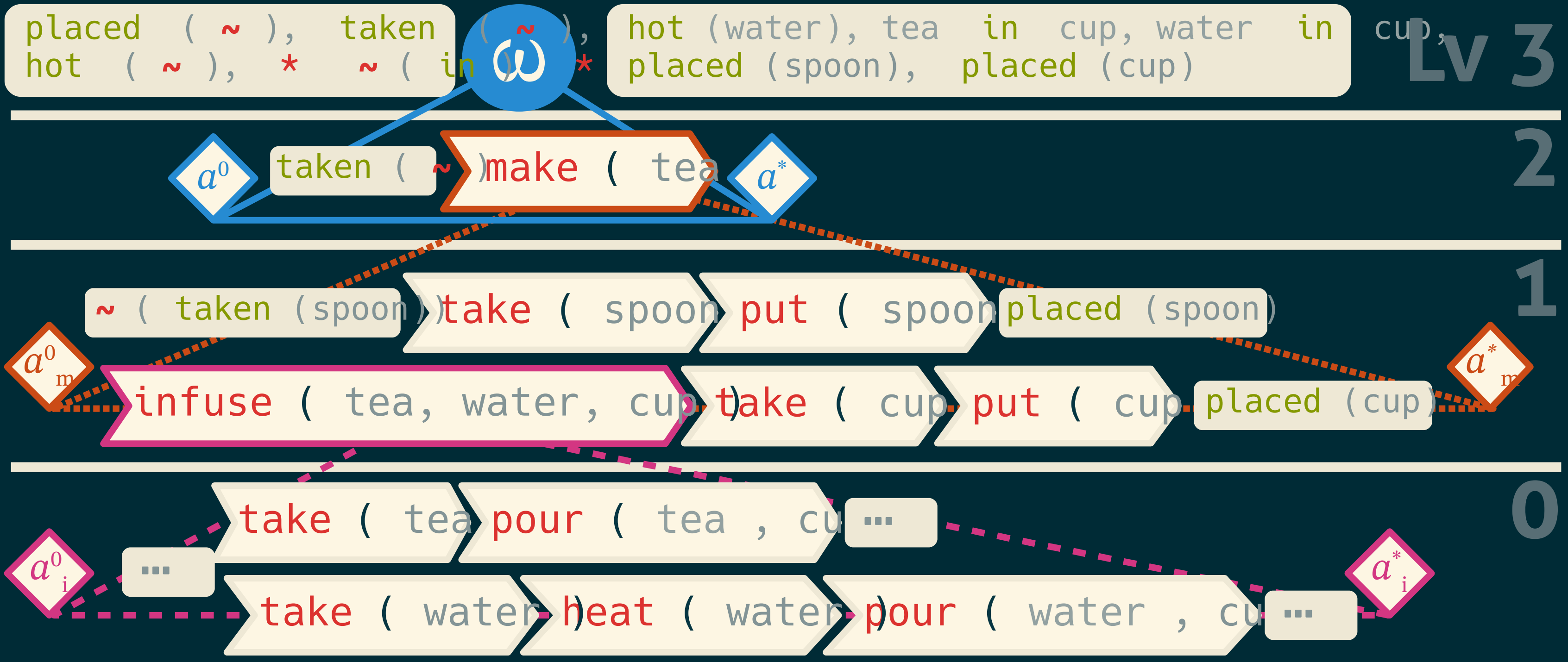
- HTN + POP planning
- Partial Resolution
  - An abstract solution at every level of abstraction
- Search by level
  - Expansion after completion :

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



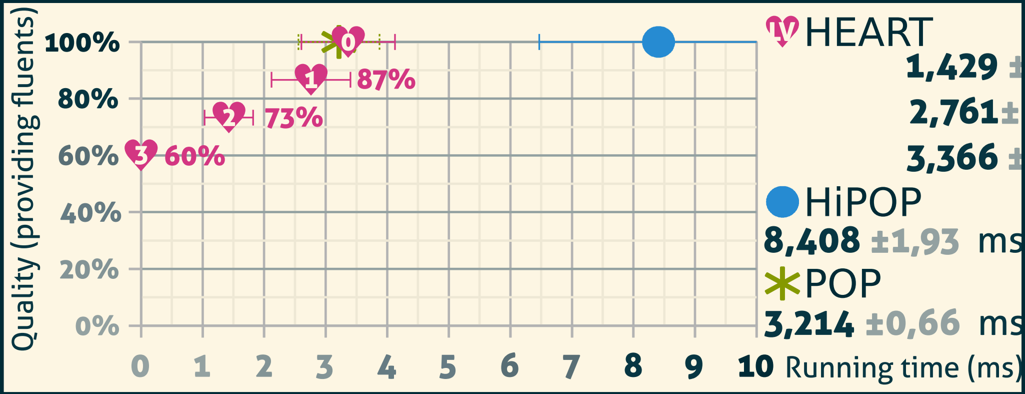
HEART

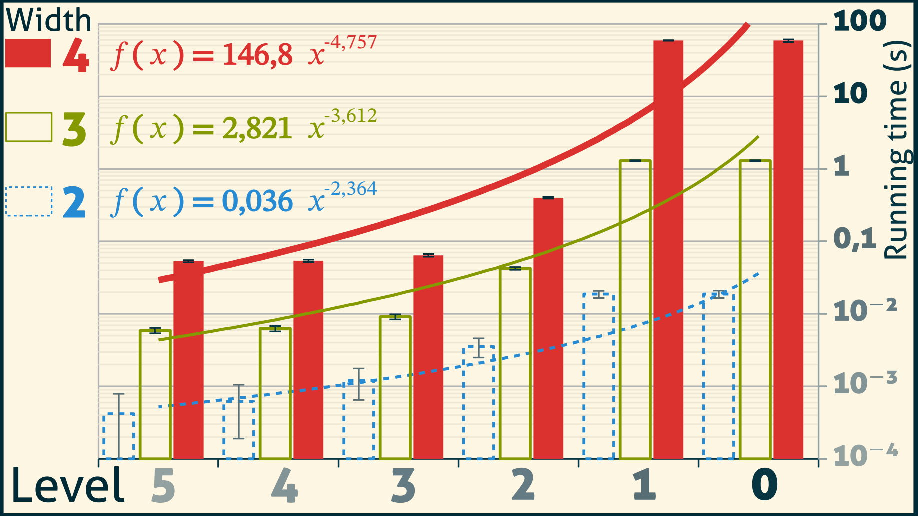
- Low priority for expansion
- Each level is a plan (abstract solution)
- Change of level
  - Propagation of atomic actions
  - Expansion of Composite Equities



Results

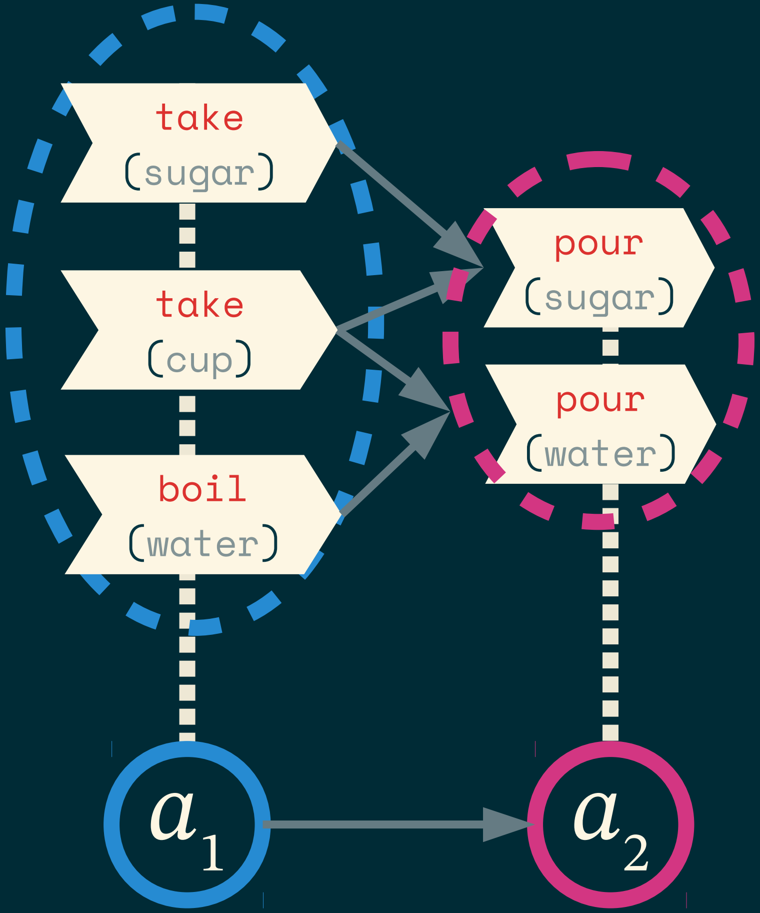
- 60% of the fluents before planning
- Exponentially faster at high abstraction levels
- Faster than HiPOP on some problems
- Common problems solved in milliseconds!





Toward Intent Recognition

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



6 Conclusion

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

Perspectives

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

Thanks for listening !

THE BEST THESIS DEFENSE IS A GOOD THESIS OFFENSE.