

Endomorphic metalanguage and abstract planning for real-time intent recognition

Antoine Gréa



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

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



- Ivan Varzinczac



- Reviewers

- Eva Onainda



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

- Damien Pellier



Endomorphic metalanguage and abstract planning for real-time intent recognition



LIRiS



Lyon 1

Antoine Gréa

1 Introduction

3

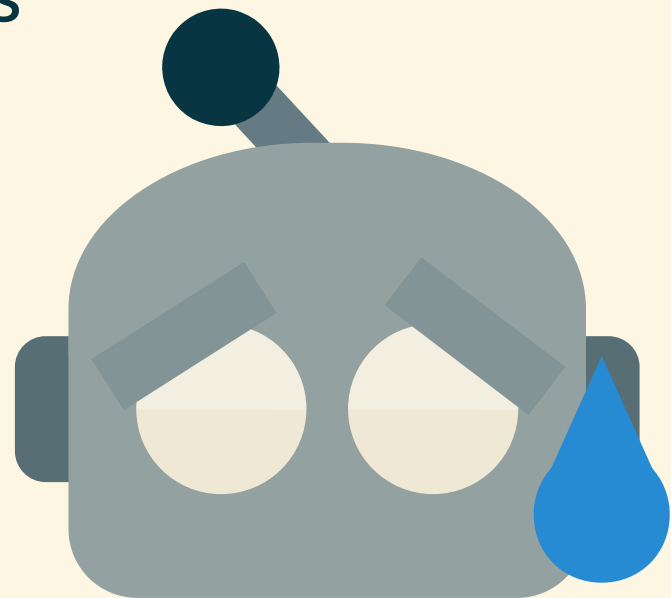


A what ?

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

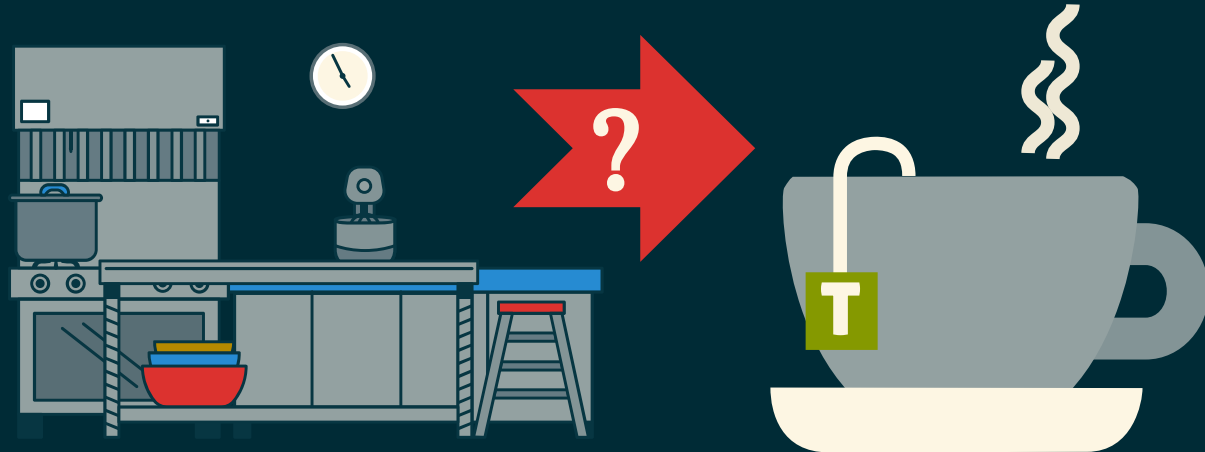
- **Intent Recognition**

- 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

6

1 Introduction

2 Intent Recognition

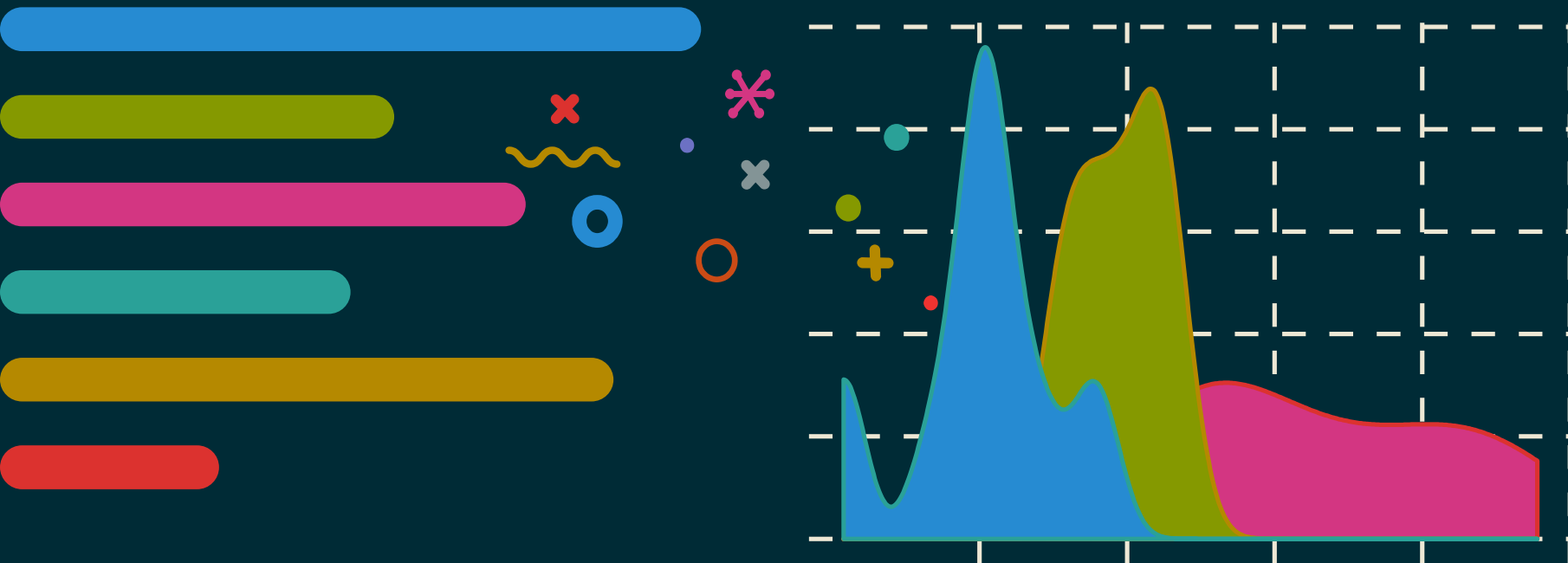
3 Knowledge Representation

4 General Planning

5 Flexible Online Planning

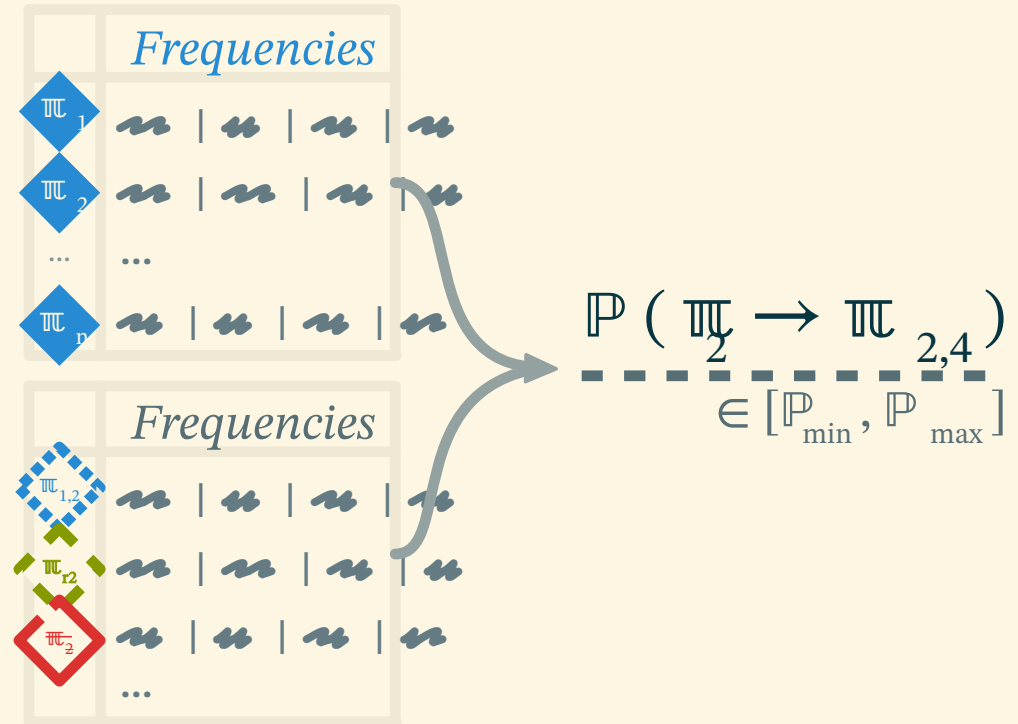
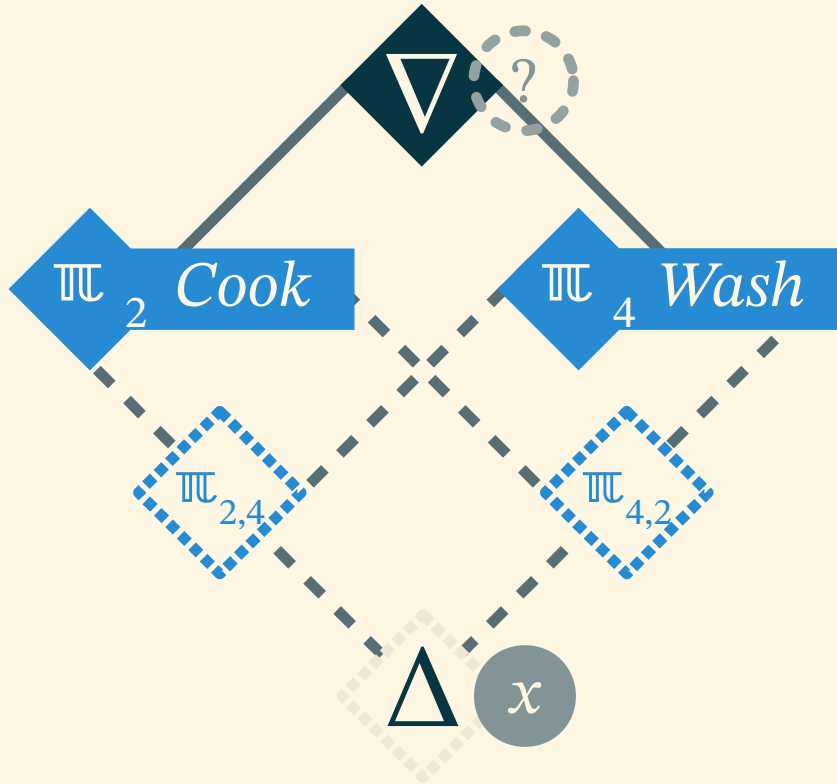
6 Conclusion

2 Intent Recognition



2.1 Logic Approach

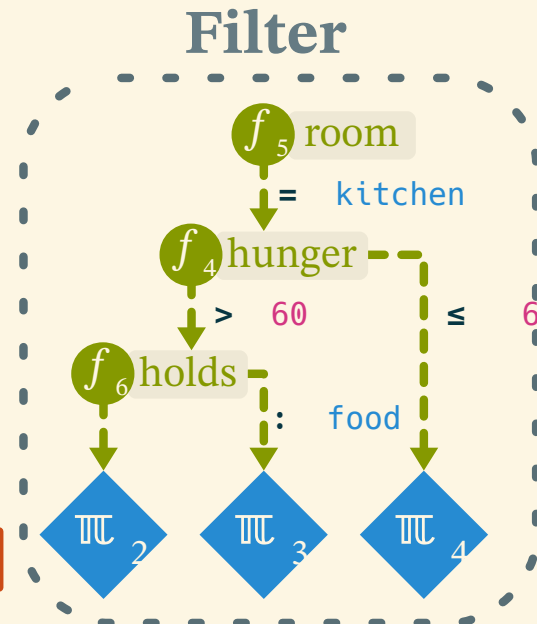
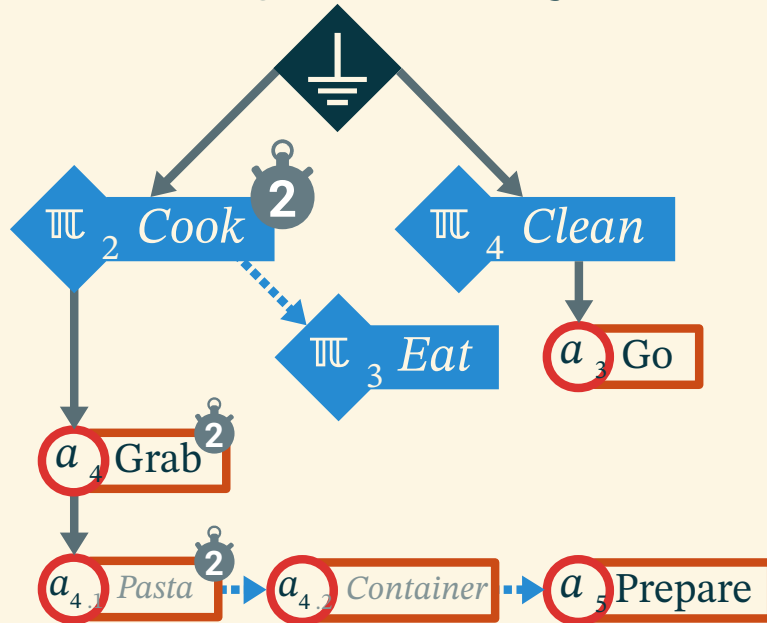
- Lattice Based : ✓ Fast computations ✗ Exponential growth



[@BOUCHARD_2006]

2.2 Stochastic Approach

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

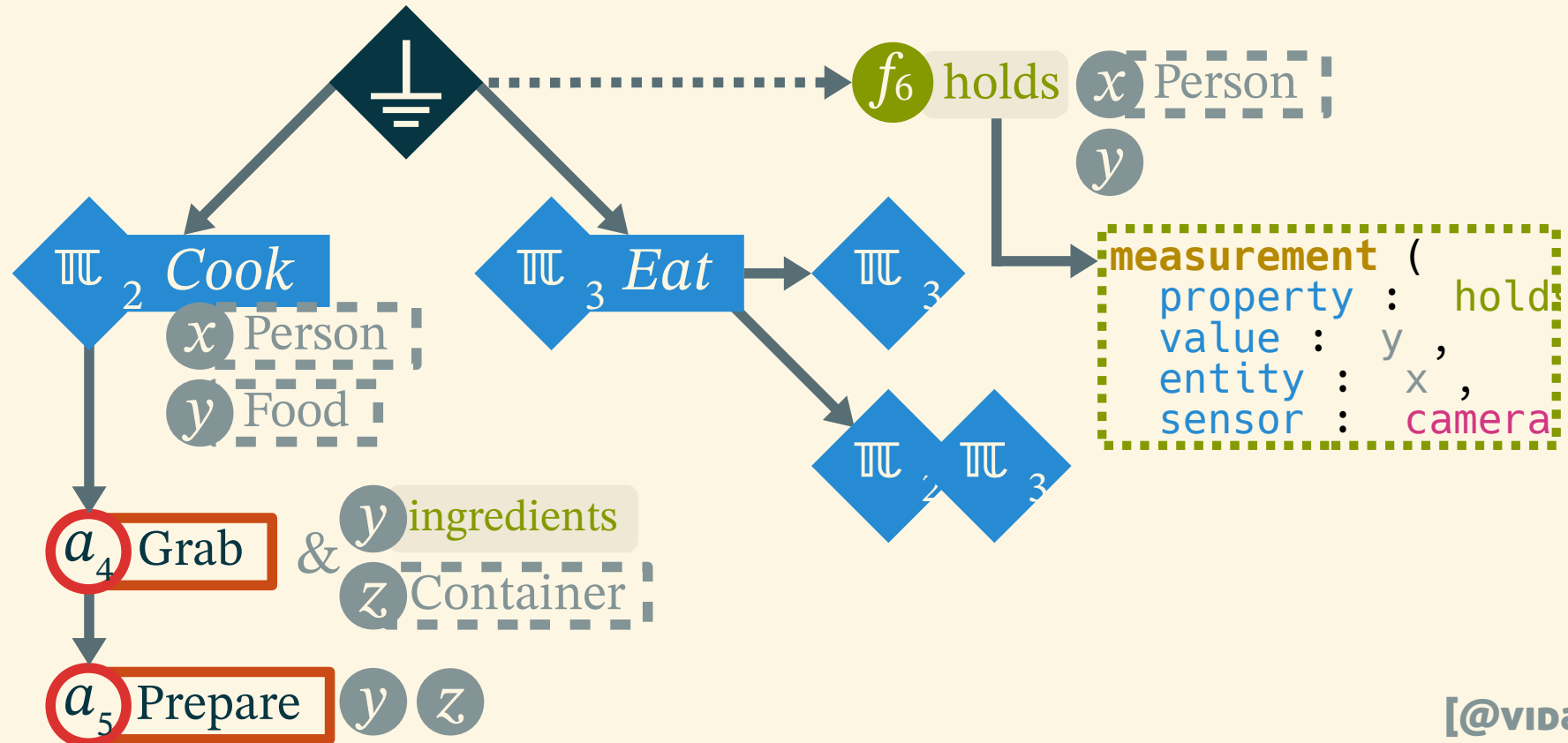


[@avrahami_2006]

2.3 Grammatical Approach

10

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



[@VIDAL_2010]

2.4 Invert Planning

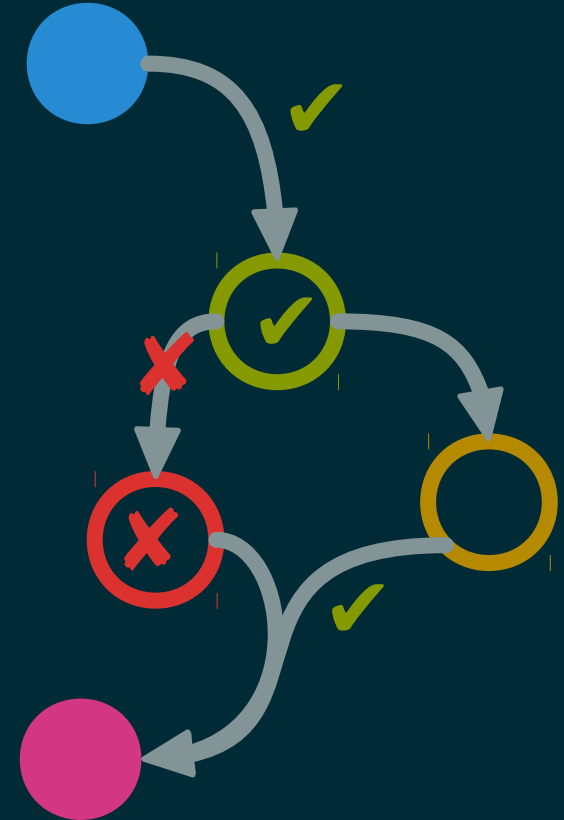
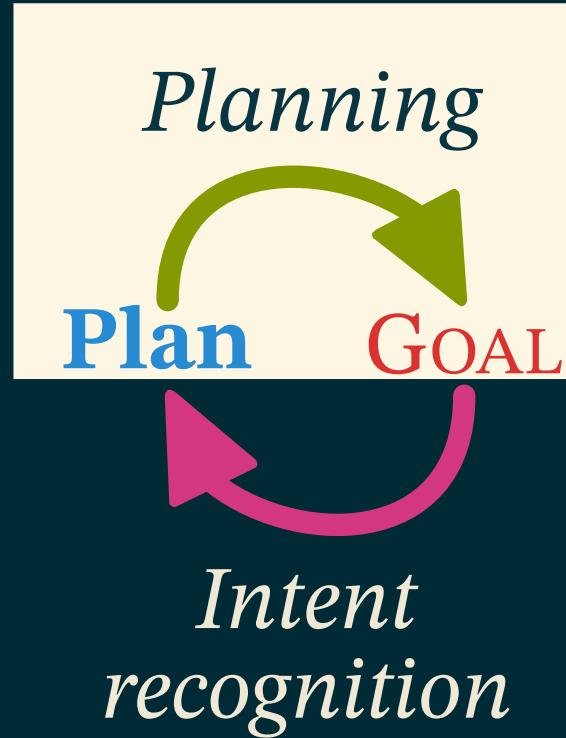
11

- Theory of Mind :

- ✓ Flexible

- ✗ More complex

“*The easier the plan,
the more likely the
goal*”



[@RAMIREZ_2008]

2.5 Framework Stacks

12

- Existing
- Contributions



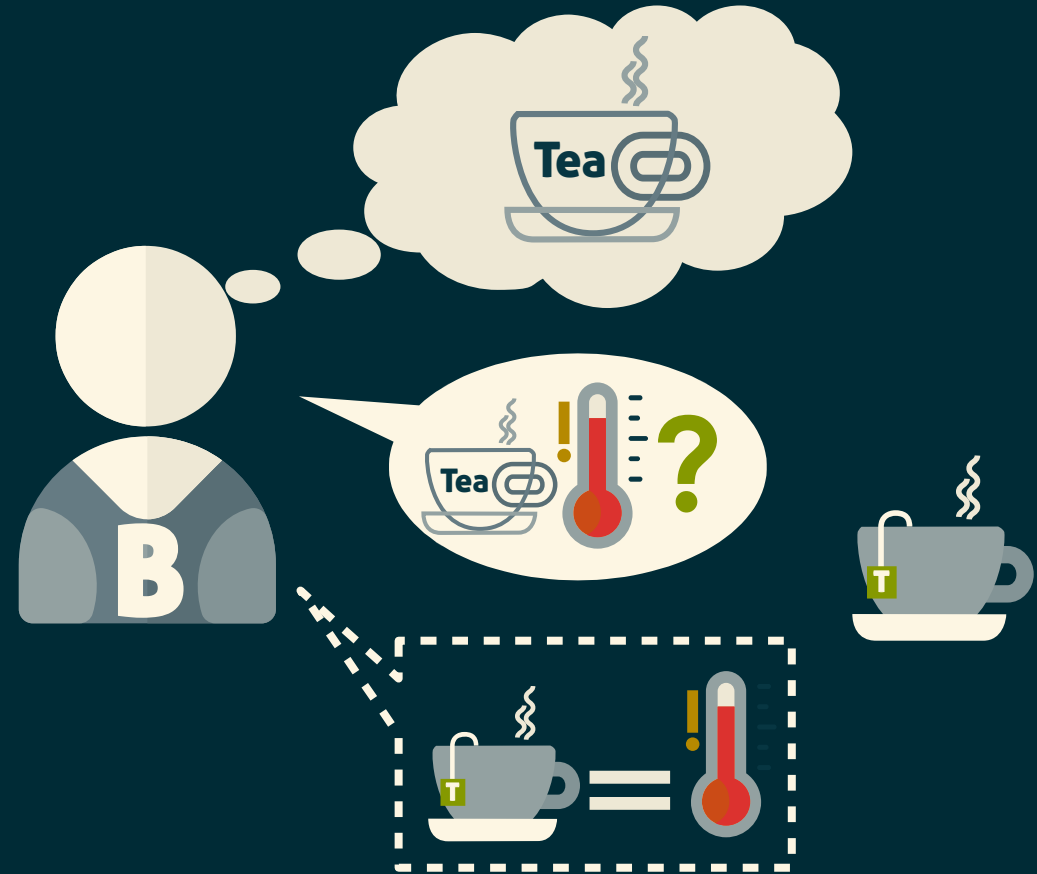
3 Knowledge Representation



How to Know

14

- **Abstraction**
 - How to **refer** to something
- **Formalization**
 - How to **talk** about something
- **Interpretation**
 - How to **know** about something



Existing Tools

- **Ontologies**

- Based on Description Logic

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

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

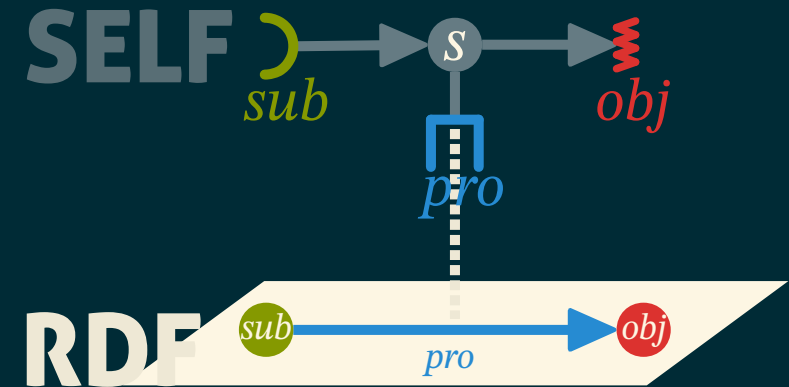
- **Languages**

- RDF
- OWL-(Lite, DL, Full)
- ...

- **Issues**

- Higher order knowledge
- Modal Logic
- Flexibility of the structure

- Defined by structure
- More expressive
- Allows complex data
- Fit for modal logic
 - Used in planning for fluents and states
 - Used in HTN for methods



Examples:

```
f = (bob @ kitchen);  
α pre f;  
α methods  
{go(kitchen) → take(cup)};
```


4 General Planning



Classical Planning

18

- 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

- Having some tea, aren't we ?

- **Fluents**

- thing taken
- hot water, tea ready



Initial State

- **Actions**

- take, brew, boil, ...

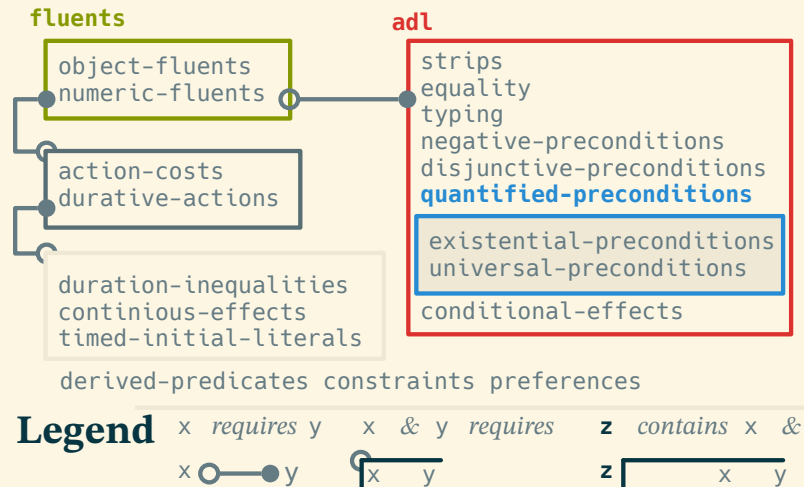


Goal State

Existing Frameworks

20

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



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

Planning Formalism Revisited

21

- States

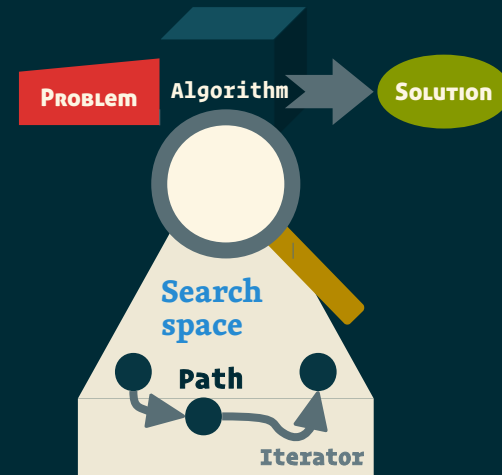
- And/Or trees of Fluents
- Verifying
- Applying

- Actions

- Precondition, Effects
- Constraints
- Cost, Duration, Probability
- Methods

- Search Space

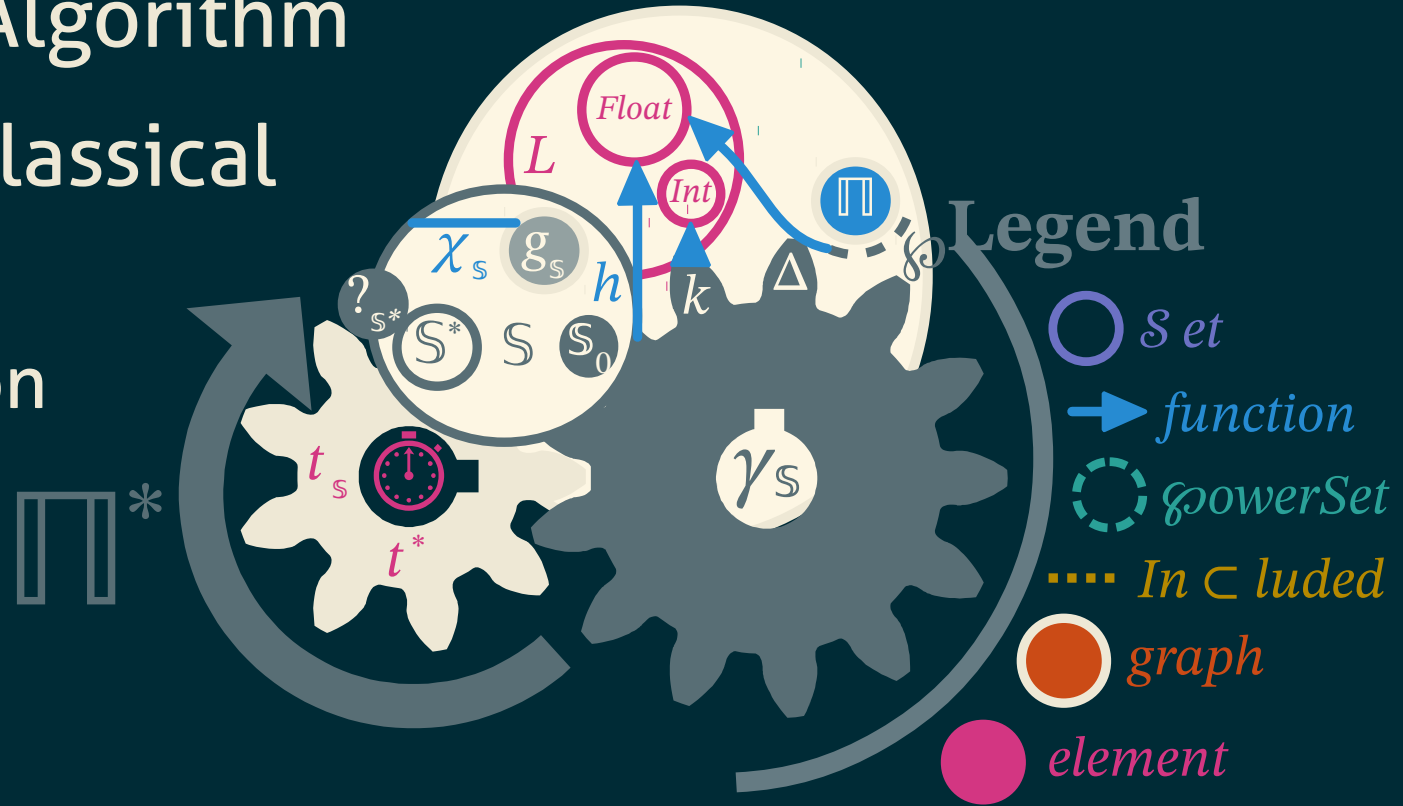
- Starting point
- Iterator
- Solution predicate



General Planning Algorithm

22

- Shortest Path Algorithm
- Instances for Classical Approaches
 - State-transition
 - Plan space
 - Case based
 - Probabilistic
 - Hierarchical



- PDDL

```
"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

24



Planning Phases

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

Domain 
compilation

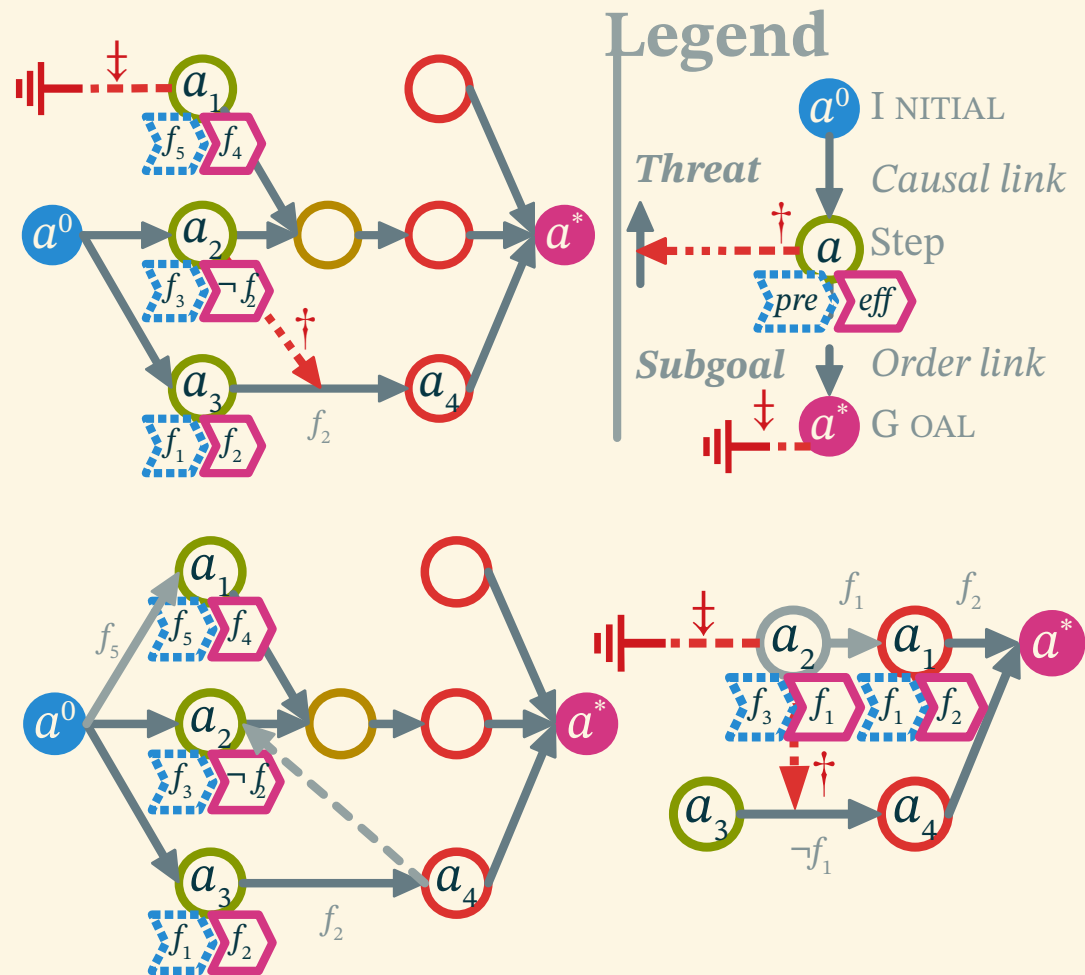
Initialisation


Planning


 Solution
optimisation

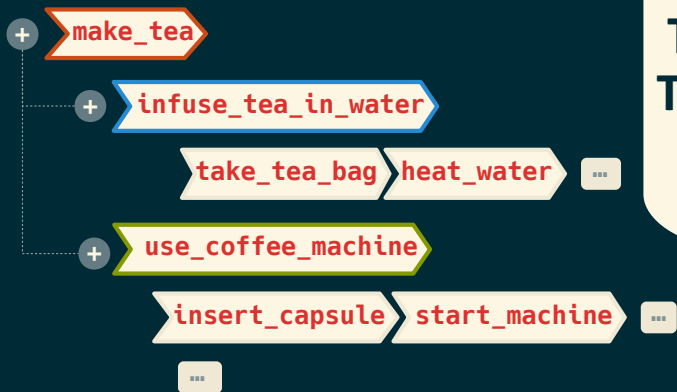
Plan Space Planning

- Exploration by refinements
- Flaws
 - Subgoals
 - Threats
- Resolvers
 - Side effects

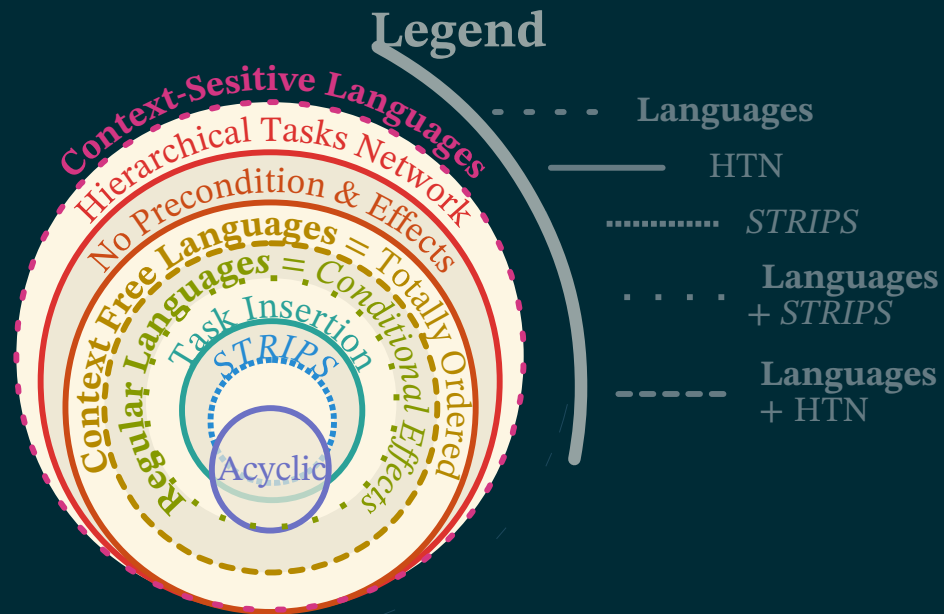


Hierarchical Task Networks

- Based on tasks
- Decomposition
- Vary in complexity



**TODO : Citation of
The difference with
planning**

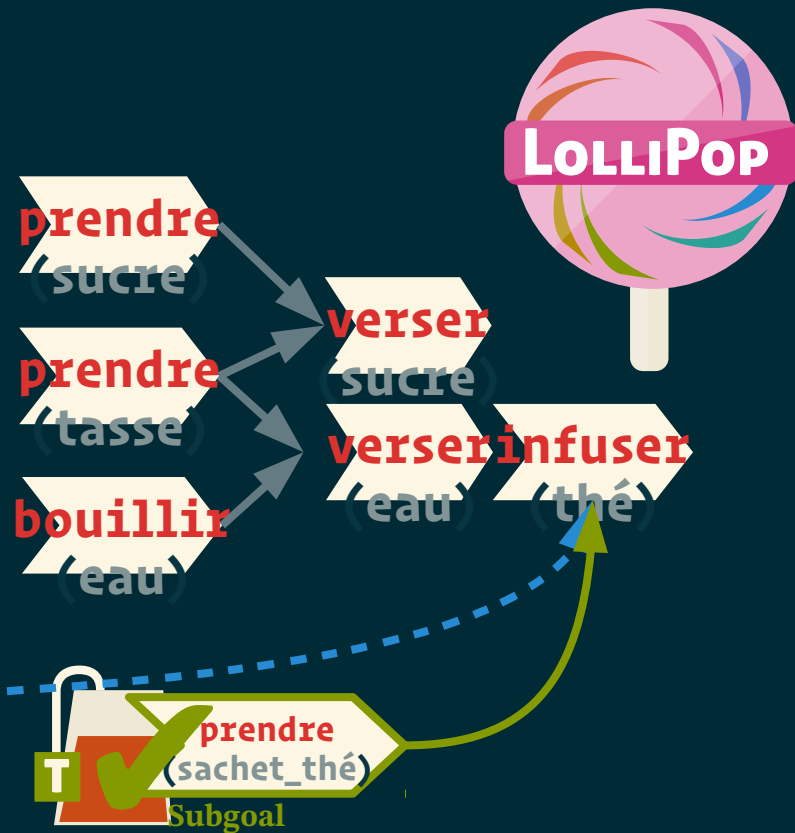


Plan Repair Prototype

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



FIXME: Font size+
English

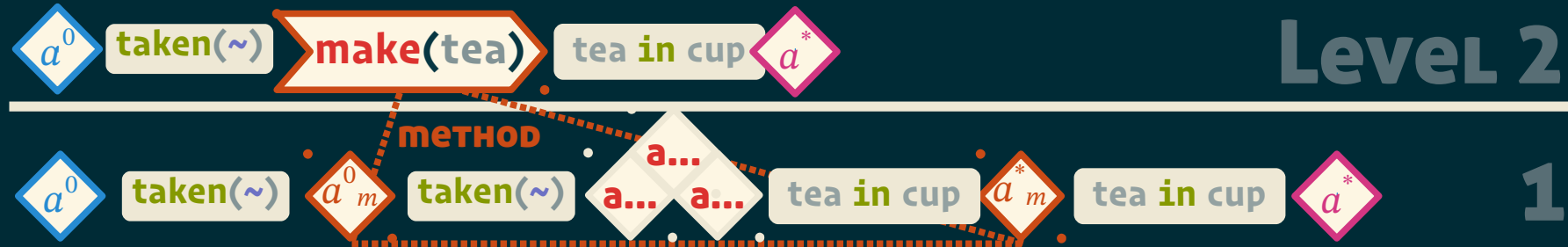


- Utility Heuristics

Abstract Planning

29

- HTN + POP planning
- Partial Resolution
 - An abstract solution at every level of abstraction
- Search by level
 - Expansion after completion :
- Decomposition flow
 - Resolver : Decompose one composite action in the plan



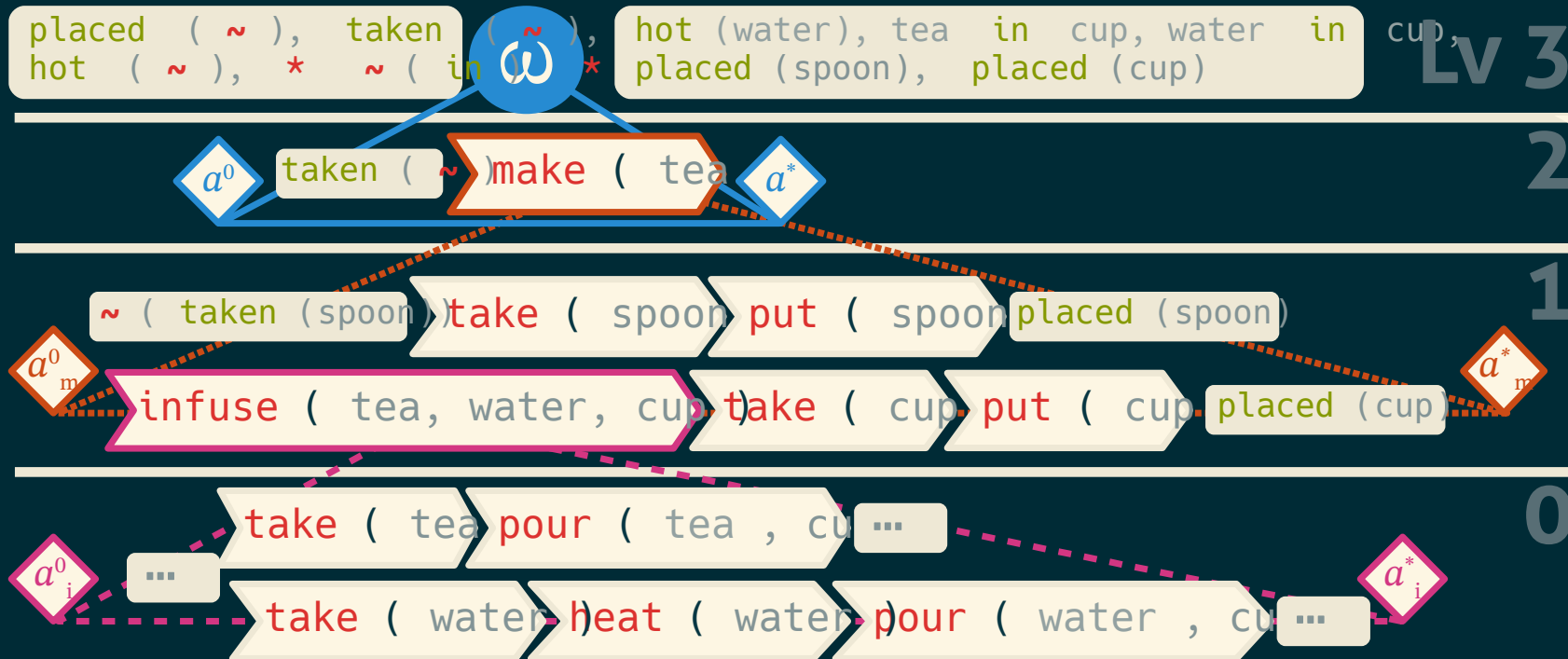
Level 2

1

HEART

30

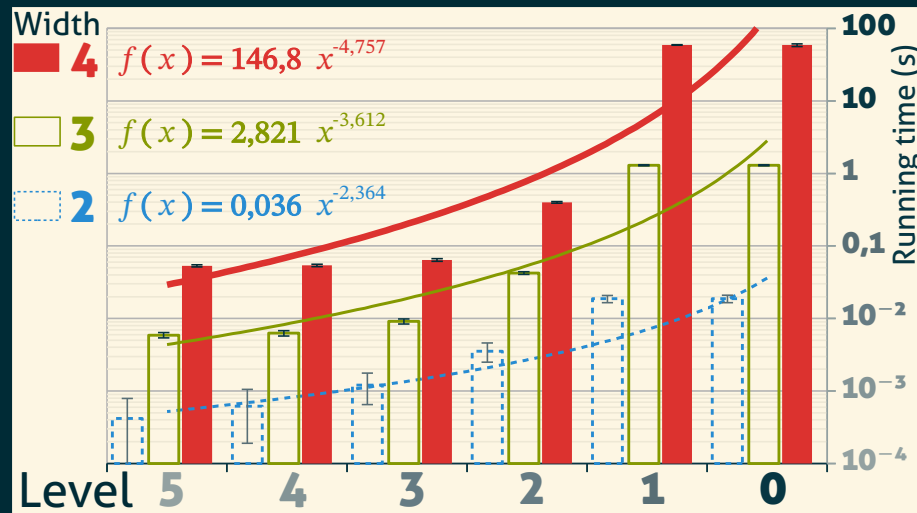
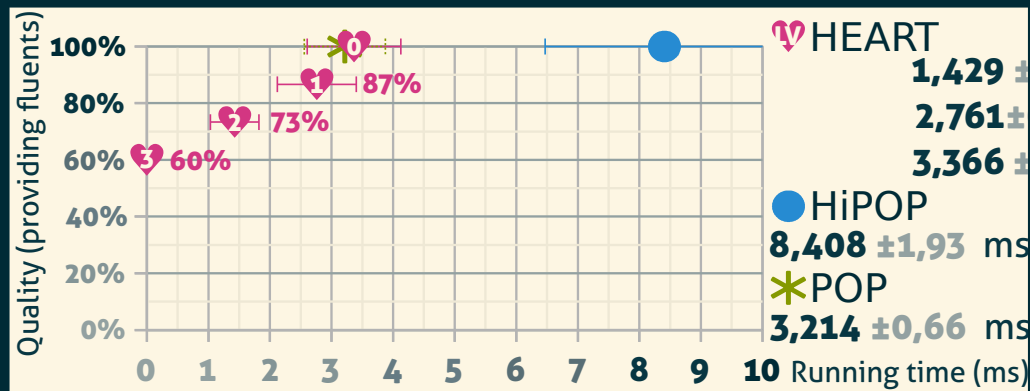
- Low priority for expansion
- Each level is a plan (abstract solution)
- Change of level
 - Propagation of atomic actions
 - Expansion of Composite Equities



TODO:
Animate for
step by step

Results

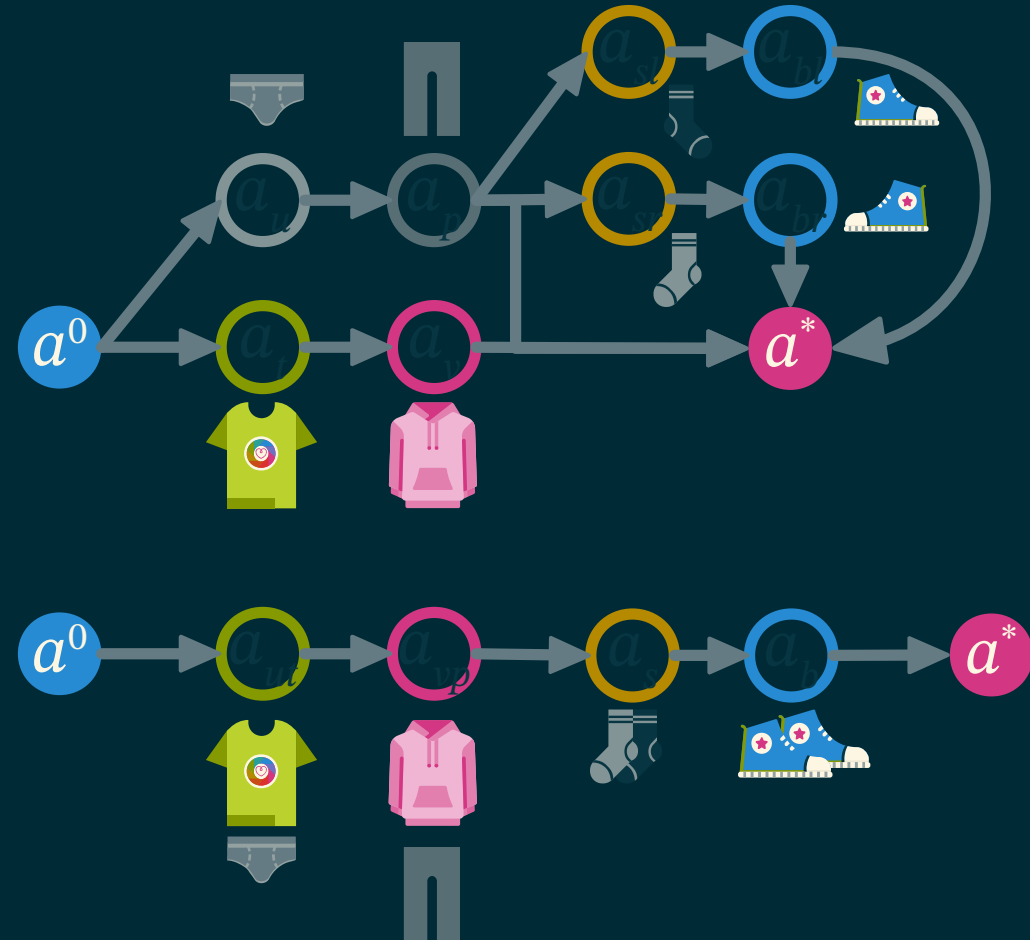
- 60% of the fluents before planning
- Exponentially faster at high abstraction levels
- Faster than HiPOP on some problems
- Common problems solved in milliseconds!



Toward Intent Recognition

32

- Linearized parallel actions
- Abstraction makes it easier
- Backward chaining is a problem



6 Conclusion

33



Contributions & Results

34

- 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
 - Simplify the instantiation workflow
 - Allow for different amount of flexibility/performances
 - Test and improve performance and queries
- 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)
 - Make a debug tool to improve domains by logging dead-ends
 - Benchmark HEART and explore heuristics

Thanks for listening !

