Agents, Planning, and Theories of Mind

Felipe Meneguzzi†

University of Aberdeen, Scotland, UK
felipe.meneguzzi@abdn.ac.uk
†Plus a lot of other people

London, September 2024

Table of Contents

Audience

Who?

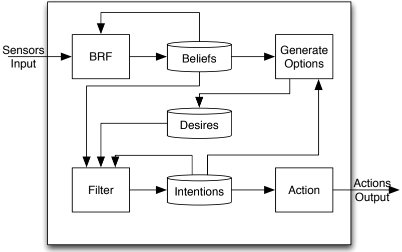
Two key demographics for my talk:

- Agents
- Planning

Agents and Planning

What?

- Agent architecture based on three "mental" structures:
  - Beliefs, Desires, and Intentions
- Based on a philosophical model for *practical reasoning*
- Provides a blueprint for agent reasoning, suitable for:
  - Agent implementations
  - Reasoning about other agents
- Key process $\rightarrow$ means ends reasoning:
  - Typically relies on a *plan library*
  - More recent work focuses on *automated planning*



```
+location(waste,X)
```

```
+location(waste,X)  :  location(robot,X)
```

```
+location(waste,X)  :  location(robot,X) & location(bin,Y)
```

```
+location(waste,X)  :  location(robot,X) & location(bin,Y)
                    <-- pick(waste);
```

```
+location(waste,X)  :  location(robot,X) & location(bin,Y)
                    <-- pick(waste);
                        !moveTo(robot,Y);
```

```
+location(waste,X)  :  location(robot,X) & location(bin,Y)
                    <-- pick(waste);
                        !moveTo(robot,Y);
                        !drop(waste).
```

```
+location(waste,X)   :  location(robot,X) & location(bin,Y)
                     <-- pick(waste);
                         !moveTo(robot,Y);
                         !drop(waste).
+!moveTo(R, To)      :  location(R, X) & X =\= To & adjacent(X, Y)
                     <-- move(X,Y);
                     <-- !+moveTo(R, To).
```

```
(:method moveTo1
    :parameters (?r - robot ?x ?y ?to - location)
    :task (moveTo ?x ?y)
    :precondition (and (location ?r ?x)
         (not (= ?x ?to)) (adjacent ?x ?y))
    :ordered-subtasks ((!move ?x ?y)
       (moveTo ?r ?to)))
)
```

- Focus of much research in AAMAS for the past three decades, primarily, on:
  - Agent Oriented Software Engineering
  - Agent reasoning cycle
  - Multiagent systems (populated by BDI agents)
- Relatively fewer efforts on the interface of means-ends reasoning and the agent model:
  - HTN Planning as lookahead:
    - de Silva, Sardiña, Padgham and others (2006-2011)
    - Patra, Nau and others (2016-): Planning and Acting, Refinement Acting Engine
    - Ingrand (2024) PROSKILL
  - State-space planners to generate new plans:
    - Meneguzzi and others (2004-)
    - Xu and Meneguzzi (2024)
    - Ingrand (2000) PROPICE

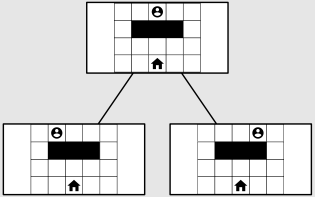
Table of Contents

Definition (Planning Task)

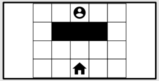
A planning task $\Pi = \langle \Xi, s_0, G \rangle$ is a tuple composed of a domain definition Ξ, an initial state s_0, and a goal state specification G. A solution to a planning task is a plan or policy π that reaches a goal state G starting from the initial state s_0 by following the transitions defined in the domain definition Ξ.

Planning problems have three key ingredients

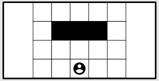
Domain Description



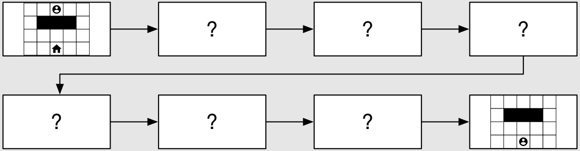
Initial State



Goal State



Solution

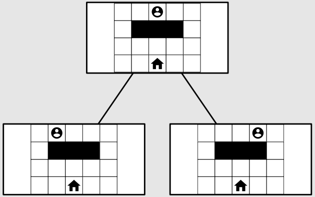


Background

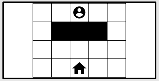
Automated Planning

Planning problems have three key ingredients

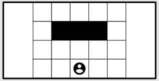
Domain Description



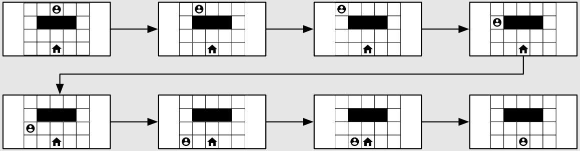
Initial State



Goal State



Solution

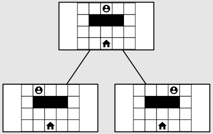


Definition (Generalised Planning Problem)

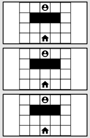
A *generalised planning problem* $\mathcal{GP} = \langle \mathcal{P}_0, \mathcal{P}_1, ..., \mathcal{P}_N \rangle$ is a set of planning problems ($N \geq 2$), where each problem $\mathcal{P}_i = \langle s_0, s_g \rangle$ shares some common structure (typically a planning domain Ξ). A solution to a generalised planning problem is a generalised plan $\Pi_{\mathcal{GP}}$, which when followed will solve any problem in $\mathcal{GP}$.

Generalised Planning problems have three key ingredients

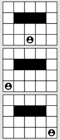
Domain Description



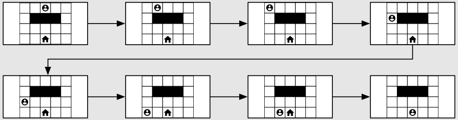
Initial States



Goal States

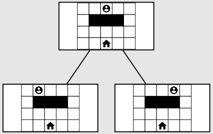


Solution

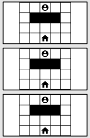


Generalised Planning problems have three key ingredients

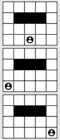
Domain Description



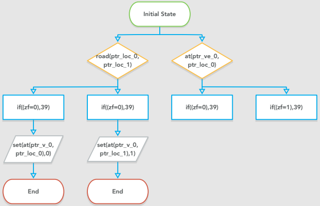
Initial States



Goal States



Solution

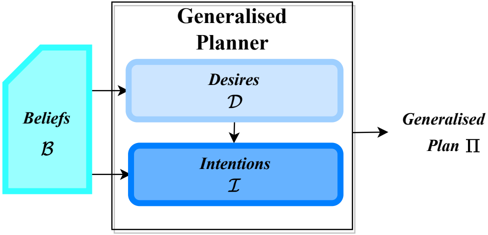


Example of a Generalised Plan

```
0. for(ptr_block_0++,9)
1. for(ptr_procunit_0++,6)
2. move(ptr_block_0,ptr_fedbelt_0,ptr_belt_0)
3. move(ptr_block_0,ptr_belt_0,ptr_procunit_0)
4. process(ptr_block_0,ptr_procunit_0)
5. move(ptr_block_0,ptr_procunit_0,ptr_belt_0)
6. endfor(ptr_procunit_0++,1)
7. move(ptr_block_0,ptr_belt_0,ptr_depbelt_0)
8. consume(ptr_block_0,ptr_depbelt_0)
9. endfor(ptr_block_0++,0)
10. end
```

Generalised Planning in BDI
Overview

- We define a high-level reasoning cycle (based on previous work[0])
  - Only declarative goals (no plan library)
  - Generalised planner
    $\rightarrow$ primary means-ends reasoning process
- Key processes:
  - Intention ($\mathcal{I}$) selection
  - Desire ($\mathcal{D}$) filtering
  - Plan Caching



[0] Felipe Meneguzzi and Lavindra de Silva. "Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning". In: *KER* 30.1 (2015), pp. 1–44.

Generalised Planning in BDI

Reasoning Cycle

procedure $\textsc{reasoningCycle}(\mathcal{B}, \mathcal{D}, \mathcal{I}, \Xi)$

  loop

    $\mathcal{B} \leftarrow \mathcal{B} \cup \textsc{sense}(\,)$

    while $\mathcal{I}$ is not empty **do**

      Pick an intention $\langle\langle\varphi, D\rangle, \pi\rangle \in \mathcal{I}$ s.t. $\mathcal{B} \models \varphi \wedge \neg D$

      $\textsc{act}(\pi)$

    Find $\{\langle\varphi_1, D_1\rangle \ldots \langle\varphi_n, D_n\rangle\} \in \mathcal{D}^2$

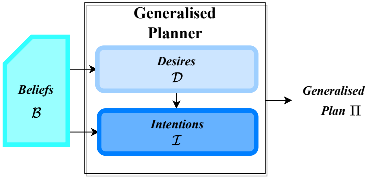
   s.t. $\exists\Pi$, $\Pi = \mathcal{G}\textsc{Planner}(\{\langle\Xi, \mathcal{B}, D_1\rangle \ldots \langle\Xi, \mathcal{B}, D_n\rangle\})$

    $\mathcal{I} \leftarrow \{\langle\langle\varphi_1, D_1\rangle, \Pi\rangle, \langle\langle\varphi_n, D_n\rangle, \Pi\rangle\}$

Generalised Planning in BDI

Key advantages

- Generalised planning problems naturally deal with concurrent desires/intentions
  - Each desire is a sub-problem in $\mathcal{GP}$
  - Resulting generalised plans $\Pi_{\mathcal{GP}}$ analogous to BDI plan-rules
  - Means-ends reasoning inherits properties of the underlying plans
- Allows us to reason about BDI agent behaviour using goal recognition
- Plan sketches allow encoding of some domain knowledge



Challenges to (Generalised) Planning in Agents

- Planning in general is expensive (Generalised Planning even more so), thus:
- Need to hedge this cost:
  - Filtering desires before invoking planner
    - Naive approach, use planning heuristics
    - Other ideas?
  - Caching plans generated at runtime
    - How to infer triggering condition?
    - Initial states might differ (even if slightly)
  - Planning offline
    - Automate plan library generation?
    - But plan for what?

Generalised Plan Sketches as a BDI Plan Library

- Recent work by Segovia-Aguas and others introduces *plan-sketches* to generalised planners
- These are incomplete plans (missing lines)
- Could be used as a plan-library of sorts

Plan Sketch Example

```
0. for(ptr_block_0++,9)
1. for(ptr_procunit_0++,6)
2. move(ptr_block_0,ptr_fedbelt_0,ptr_belt_0)
3. empty
4. empty
5. empty
6. endfor(ptr_procunit_0++,1)
7. move(ptr_block_0,ptr_belt_0,ptr_depbelt_0)
8. consume(ptr_block_0,ptr_depbelt_0)
9. endfor(ptr_block_0++,0)
10. end
```

Plan Sketch Example

```
0. for(ptr_block_0++,9)
1. for(ptr_procunit_0++,6)
2. move(ptr_block_0,ptr_fedbelt_0,ptr_belt_0)
3. move(ptr_block_0,ptr_belt_0,ptr_procunit_0)
4. process(ptr_block_0,ptr_procunit_0)
5. move(ptr_block_0,ptr_procunit_0,ptr_belt_0)
6. endfor(ptr_procunit_0++,1)
7. move(ptr_block_0,ptr_belt_0,ptr_depbelt_0)
8. consume(ptr_block_0,ptr_depbelt_0)
9. endfor(ptr_block_0++,0)
10. end
```

Generalised Plan Sketches as a BDI Plan Library

- Recent work by Segovia-Aguas and others introduces *plan-sketches* to generalised planners
- These are incomplete plans (missing lines)
- Could be used as a plan-library of sorts

Generalised Plan Sketches as a BDI Plan Library

- Recent work by Segovia-Aguas and others introduces *plan-sketches* to generalised planners
- These are incomplete plans (missing lines)
- Could be used as a plan-library of sorts
  But how to generate them?
- Potential solutions:
  - Naive solution: manual generation (another flavour of AOSE)
  - Caching previously generated plans (open research question)

Table of Contents

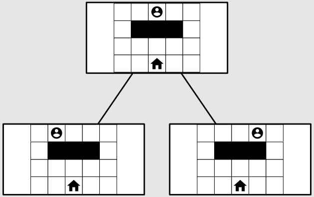
Definition (Goal Recognition Task)

A goal recognition task $\Pi_{\mathcal{G}}^{\Omega_\pi} = \langle \Xi, s_0, \mathcal{G}, \Omega_\pi \rangle$ is a tuple composed of a domain definition Ξ, an initial state s_0, a set of goal hypotheses $\mathcal{G}$, and a sequence of observations Ω_π.

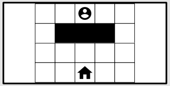
Background
Goal Recognition

Goal/Plan Recognition problems have **four** key ingredients

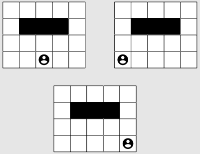
Domain Description



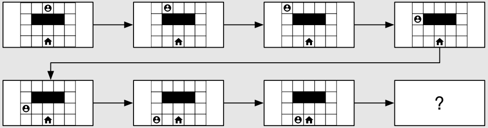
Initial State



Goal **Hypotheses**



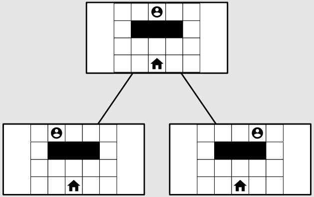
Observations



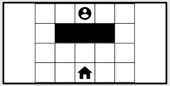
Background
Goal Recognition

Goal/Plan Recognition problems have **four** key ingredients

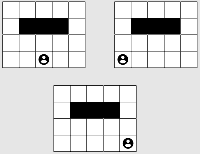
Domain Description



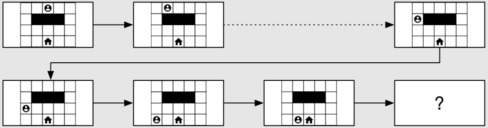
Initial State



Goal **Hypotheses**

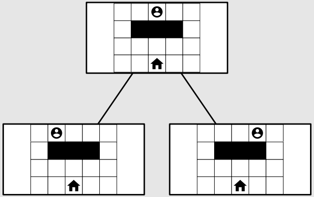


Observations

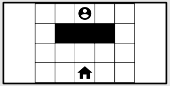


Goal/Plan Recognition problems have **four** key ingredients

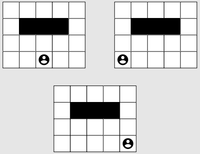
Domain Description



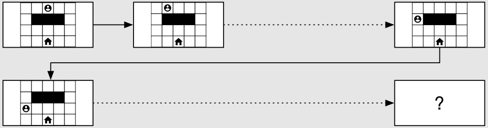
Initial State



Goal **Hypotheses**

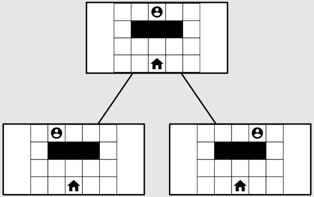


Observations

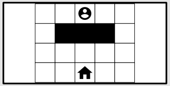


Goal/Plan Recognition problems have **four** key ingredients

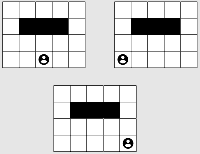
Domain Description



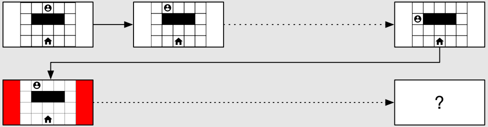
Initial State



Goal **Hypotheses**

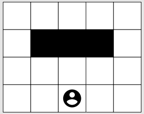


Observations



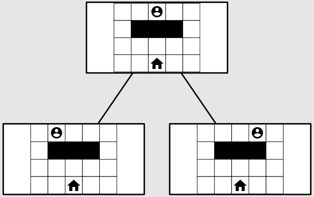
Solution

Correct Goal

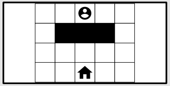


Goal/Plan Recognition problems have **four** key ingredients

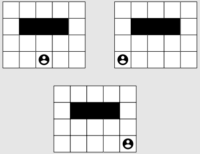
Domain Description



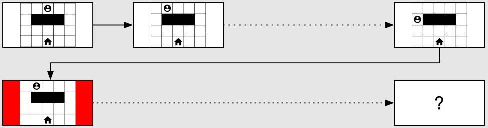
Initial State



Goal **Hypotheses**

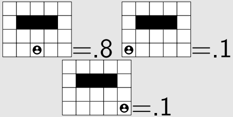


Observations



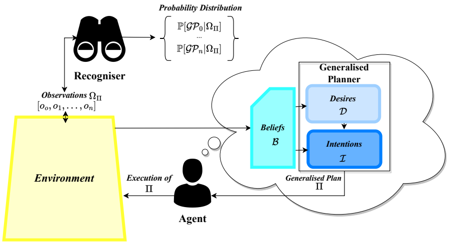
Solution

Probability Distribution



- We define a generalised goal recognition problem $\langle \mathbb{G}, \Omega_\Pi \rangle$, where $\mathbb{G} = \langle \mathcal{GP}_0, \mathcal{GP}_1, ..., \mathcal{GP}_N \rangle$
- Solving this problem consists of computing posterior probabilities over $\mathbb{G}$ given Ω_Π:

$$\mathbb{P}(\mathcal{GP} \mid \Omega_\Pi) = \eta * \mathbb{P}(\Omega_\Pi \mid \mathcal{GP}) * \mathbb{P}(\mathcal{GP})$$



- We define a generalised goal recognition problem $\langle \mathbb{G}, \Omega_\Pi \rangle$, where $\mathbb{G} = \langle \mathcal{GP}_0, \mathcal{GP}_1, ..., \mathcal{GP}_N \rangle$
- Solving this problem consists of computing posterior probabilities over $\mathbb{G}$ given Ω_Π:

$$\mathbb{P}(\mathcal{GP} \mid \Omega_\Pi) = \eta * \mathbb{P}(\Omega_\Pi \mid \mathcal{GP}) * \mathbb{P}(\mathcal{GP})$$

- BDI reasoning cycle and goal recognition provide an effective *Theory of Mind*

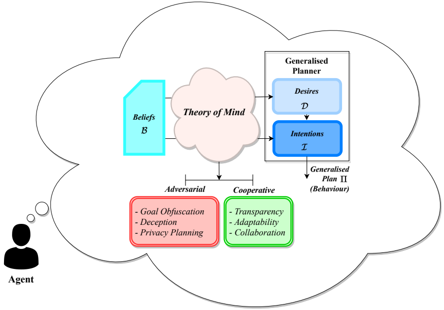


Table of Contents

Promising Approaches

Moving forward

BDI agents using our model now have a model and an inference mechanism to be fully aware of others:

- Adversarial Contexts (counterplanning)
- Cooperative Contexts (transparent planning)

Challenges and Opportunities

- This paper lays out a generic framework, but most of its components are still open research questions
  - Reasoning cycle
    - computationally expensive
    - high-level, no failures, no replaning
  - Generalised recognition approaches are still in their infancy
- However, this provides a research agenda for many years to come
- Stuff I did not discuss (sorry)
  - SOAR (Laird and others)
  - Reinforcement Learning as a Means-Ends reasoner

Acknowledgement

- Ramon Pereira (University of Manchester)
- Nir Oren (University of Aberdeen)
- Sebastián Sardiña (RMIT)
- Javier Segovia-Aguas (Universitat Pompeu Fabra)
- Lavindra de Silva (Cambridge University)
- André Pereira (UFRGS)
- Miquel Ramirez (University of Melbourne)
- Reuth Mirsky (Bar-Ilan University)
- Leonardo Amado (University of Aberdeen)
- Michael Luck (University of Sussex)

And Sanjay Modgil (KCL)