# 7 Toward Intent Recognition

Oui mais surtout dire le lien avec les chapitres précédents !

Since the original goal of this thesis was on intent recognition of dependent persons, we need to explain some more about that specific domain. The problem is to infer the goals of an external agent through only observations without intervention. In the end, the idea is to infer that goal confidently enough to start assisting the other agent without explicit instructions.

## 7.1 Domain problems

donner refs ou dire si c'est présenté plus tard dans ton chapitre

This field was widely studied. Indeed, at the end of the last century, several works started using *abduction* to infer intents from observational data. Of course this comes as a challenge since there is a lot of uncertainty involved. We have no reliable information on the possible goals of the other agent and we don't have the same knowledge of the world. Also, observations can sometimes be incomplete or misleading, the agent can abandon goals or pursue several goals at once while also doing them sub-optimally or even fail at them. To finish, sometimes agents can actively try to hide their intents to the viewer.

### 7.1.1 Observations and inferences

As explained above, we can only get close to reality and any progress in relevance and detail is exponentially expensive in terms of computing and memory resources. That is why any system will maintain a high degree of abstraction that will cause errors inherent in this approximation.

This noise phenomenon can impact the activity and situation recognition system and therefore seriously impact the intention recognition and decision-making system with an amplification of the error as it is processed. It is also important to remember that this phenomenon of data noise is also present in inhibition and that the lack of perception of an event is as disabling as the perception of the wrong event.

It is possible to protect recognition systems in an appropriate way, but this often implies a restriction on the levels of possibilities offered by the system such as specialized recognition or recognition at a lower level of relevance.

### 7.1.2 Cognitive inconsistencies

In the field of personal assistance, activity recognition is a crucial element. However, it happens that these events are very difficult to recognize. The data noise mentioned above can easily be confused with an omission on the part of the observed agent. This dilemma is also present during periods of inactivity, the system can start creating events from scratch to fill what it may perceive as inhibition noise.

These problems are accompanied by others related to the behavior of the observed agent. For example, they may perform unnecessary steps, restart ongoing activities or suddenly abandon them. It is added that other aspects of observations can make automated inferences such as ambiguous actions or the agent performing an action that resembles another complicated.

However, some noise problems can be easily detected by simple cognitive processes such as impossible sequences (e. g. closing a closed door). Contextual analyses provide a partial solution to some of these problems.

### 7.1.3 Sequentiality

Since our recognition is based on a highly temporal planning aspect, we must take into account the classic problems of sequentiality.

A first problem is to determine the end of one plan and the beginning of another. Indeed, it is possible that some transitions between two planes may appear to be a plane in itself and therefore may cause false positives. Another problem is that of intertwined planes. A person can do two things at once, such as answering the phone while cooking. An action in an intertwined plan can then be identified as a discontinuation of activity or a logical inconsistency. A final problem is that of overloaded actions. Not only can an agent perform two tasks simultaneously, but also perform an action that contributes to two activities. These overloaded actions make the process of intention recognition complex because they are close to data noise.

## 7.2 Existing approaches

The problem of intention recognition has been strongly addressed from many angles. It is therefore not surprising that there are many paradigms in the field. The first studies on the subject highlight the fact that intention recognition problems are problems of abductive logic or graph coverage. Since then, many models have competed in imagination and innovation to improve the field. These include constraint system-based models that provide a solution based on pre-established rules and compiled plan libraries, those that use state or action networks that then launch algorithms on this data, and reverse planning systems.

### 7.2.1 Constraint

One of the approaches to intention recognition is the one that builds a system around a strong logical constraint. There is often a time constraint system that is complemented by various extensions to cover as many sequential problems as possible.

#### 7.2.1.1 Deductive Approach

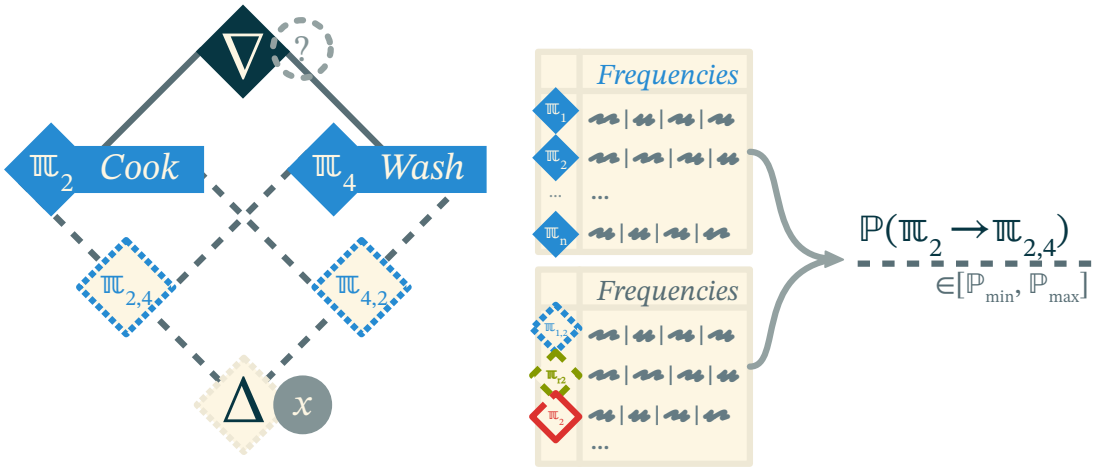
In order to solve a problem of intention recognition, abductive logic can be used. Contrary to deductive logic, the goal is to determine the objective from the observed actions. Among the first models introduced is Goldman *et al.* (1999)'s model, which uses the principle of action to construct a logical representation of the problem. This paradigm consists in creating logical rules as if the action in question was actually carried out, but in hypothesizing the predicates that concretize the action and thus being able to browse the research space thus created in order to find all the possible plans containing the observed actions and concretizing defined intentions. This model is strongly based on first-order logic and SWI Prolog logic programming languages. Although revolutionary for the time, this system pale in comparison to recent systems, particularly in terms of prediction accuracy.

? pales ?

#### 7.2.1.2 Algebraic Approach

Some paradigms use algebra to determine possible plans from observed actions. In particular, we find the model of Bouchard *et al.* (2006) which extends the subsumption relationship from domain theory to the description of action and sequence of action in order to introduce it as an order relationship in the context of the construction of a lattice composed of possible plans considering the observed actions. This model simply takes into account the observed actions and selects any plan from the library of plans that contains at least one observed action. Then this paradigm will construct all the possible plans that correspond to the Cartesian product of the observed actions with the actions contained in the selected plans (while respecting their order). This system makes it possible to obtain a subsumption relationship that corresponds to the fact that the plans are more or less general. Unfortunately, this relationship alone does not provide any information on which plan is most likely.

That is why Roy *et al.* (2011) created a probabilistic extension of this model. This uses frequency data from a system learning period to calculate the influence probabilities of each plane in the recognition space. This makes it possible to calculate probabilistic intervals for each plan, action as well as for a plan to know a given action. In order to determine the probability of each plane knowing the upper bound of the lattice (plane containing all observed actions) the sum of the conditional probabilities of the plane for each observed action divided by the number of observed actions is made. This gives a probability interval for each plane allowing the ordinates. This model has the advantage of considering many possible plans but has the disadvantage of seeing a computational explosion as soon as the number of observed actions increases and the context is not taken into account.




citer la figure et expliquer la gifure (les différents éléments), l'exemple que utilises, la méthode représentée … !!! on ne comprends pas !

Figure 7.1: The lattice formed by observations (top), matching plans, possible hypothesis and problem (bottom)

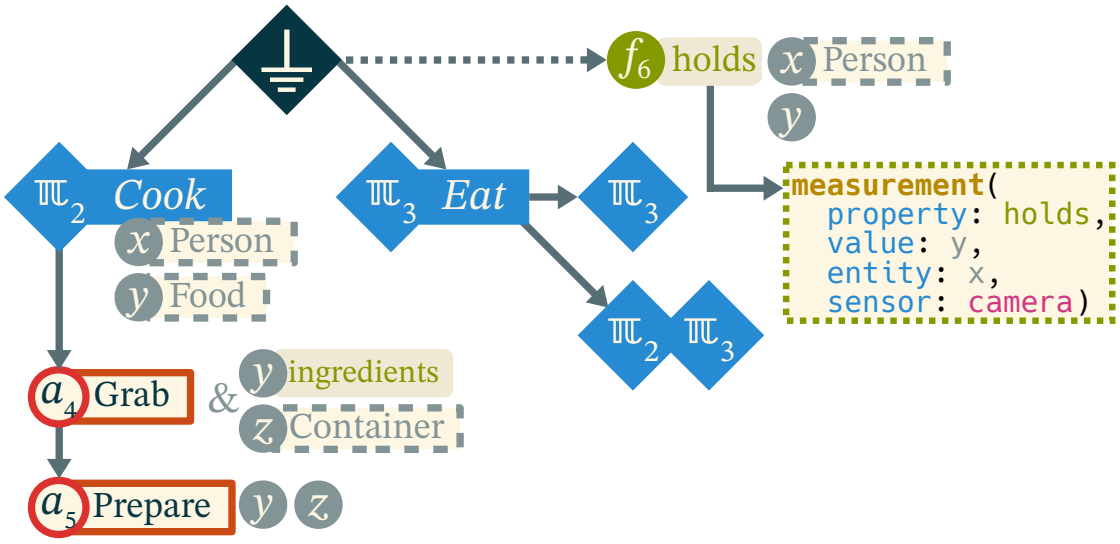
#### 7.2.1.3 Grammatical Approach

Another approach is that of grammar. Indeed, we can consider actions as words and sequences as sentences and thus define a system that allows us to recognize shots from incomplete sequences. Vidal *et al.* (2010) has therefore created a system of intention recognition based on grammar. It uses the evaluated grammar system to specify measurements from observations. These measures will make it possible to select specific plans and thus return a hierarchical hypothesis tree with the actions already carried out, the future and the plans from which they are derived. This model is very similar to first order logic-based systems, and uses a SWI Prolog type logic language programming system. Given the scope of maritime surveillance, this model, although taking very well into account the context and the evolution of the measures, is only poorly adapted to an application in assistance, particularly in the absence of a system for discriminating against results plans.

#### 7.2.1.4 Linear programming approach

Another class of approaches is that of diverting standard problem-solving tools to solve the problem of intention recognition. It is therefore possible, by modifying traditional algorithms or by transforming a problem, to ensure that the solution of the tool corresponds to the one sought.

Inoue and Inui (2011) develops the idea of a model that uses linear programming to solve the recognition problem. Indeed, observations are introduced in the form of causes in relation to hypotheses, in a first-order logic predicate system. Each atom is then weighed and introduced into a process of problem transformation by feedback and the introduction of order and causality constraints in order to force the linear program towards optimal solutions by taking into account observations. Although ingenious, this solution does not discriminate between possible plans and is very difficult




idem que 7.1

Figure 7.2: The valued grammar used for intent recognition

to apply to real-time recognition situations, mainly because of the problem transformation procedure required each time the problem is updated.

#### 7.2.1.5 Markovian Logic Approach

manque l'année

Another constraint paradigm is the one presented by Raghavana *et al.* using a Markovian extension of first-order logic. The model consists of a library of plans represented in the form of Horn clauses indicating which actions imply which intentions. The aim is therefore to reverse the implications in order to transform the deduction mechanism into an abduction mechanism. Exclusionary constraints and a system of weights acquired through learning must then be introduced to determine the most likely intention. Once again, despite the presence of a system of result discrimination, there is no consideration of context and abductive transformation remains too cumbersome a process for real-time recognition.

implIES

remains A too

### 7.2.2 Networks

#### 7.2.2.1 And/Or trees approach

As in its early days, intention recognition can still be modeled in the form of graphs. Very often in intention recognition, trees are used to exploit the advantages of acyclicity in resolution and path algorithms. In the prolific literature of Geib et al. we find the model at the basis of PHATT (Geib 2002) which consists of an AND/OR tree representing a HTN that contains the intentions as well as their plans or methods. A prior relationship is added to this model and it is through this model that constraints are placed on the execution of actions. Once an action is observed, all the successors of

the action are unlocked as potential next observed action. We can therefore infer by hierarchical path the candidate intentions for the observed sequence.

Since this model does not allow discrimination of results, Geib and Goldman (2005) then adds probabilities to the explanations of the observations. The degree of coverage of each possible goal is used to calculate the probability of each goal. That is, the goal with the plan containing the most observed action and the least unobserved action will be the most likely. This is very ingenious, as the coverage rate is one of the most reliable indicators. However, the model only takes into account temporality and therefore has no contextual support. The representation in the form of a tree also makes it very difficult to be flexible in terms of the plans, which are then fixed a priori.

#### 7.2.2.2 HTN Approach

The HTN model is often used in the field, such as the hierarchical tree form used by Avrahami-Zilberbrand *et al.* (2005). The tree consists of nodes that represent various levels of action and intent. A hierarchical relationship links these elements together to define each intention and its methods. To this tree is added an anteriority relationship that constrains the execution order. This paradigm uses time markers that guarantee order to use an actualization algorithm that also updates a hypothesis tree containing possible intentions for each observation.

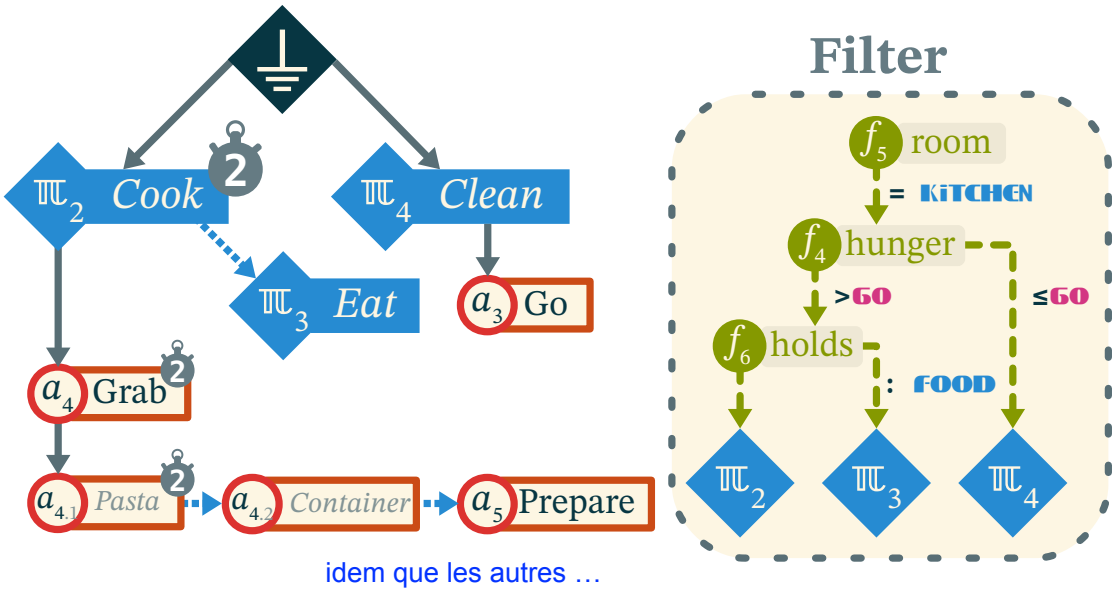



Figure 7.3: The HTN and decision tree used for intent recognition

A probabilistic extension of the Avrahami-Zilberbrand and Kaminka (2006) applies a hierarchical hidden Markov model to the action tree. Using three types of probability that of plan tracking, execution interleaving and interruption, we can calculate the probability of execution of each plan according to the observed sequence. The logic and contextual model filtered on the possible plans upstream leaving us with few calculations to order these plans.

This contextual model uses a decision tree based on a system of world properties. Each property has a finite (and if possible very limited) number of possible values. This allows you to create a tree containing for each node a property and an arc for each value. This is combined with other nodes or leaves that are actions. While running through the tree during execution, the branches that do not correspond to the current value of each property are pruned. Once a leaf is reached, it is stored as a possible action. This considerably reduces the research space but requires a balanced tree that is not too large or restrictive.

#### 7.2.2.3 Hidden Markovian Approach

When we approach stochastic models, we very often find Markovian or Bayesian models. These models use different probabilistic tools ranging from simple probabilistic inference to the fusion of stochastic networks. It can be noted that probabilities are often defined by standard distributions or are isomorphic to weighted systems.

A stochastic model based on THRs is the one presented by Blaylock and Allen (2006). This creates hierarchical stacks to categorize abstraction levels from basic actions to high level intentions. By chaining a hidden Markov model to these stacks, the model is able to affect a probability of intention according to the observed action.

#### 7.2.2.4 Bayesian Approach

Another stochastic paradigm is the one of Han and Pereira (2013). It uses Bayesian networks to define relationships between causes, intentions and actions in a given field. Each category is treated separately in order to reduce the search space. The observed actions are then selected from the action network and extracted. The system then uses the intention network to build a temporary Bayesian network using the NoisyOR method. The network created is combined in the same way with the network of causes and makes it possible to have the intention as well as the most probable cause according to the observations.

#### 7.2.2.5 Markovian network approach

The model of Kelley *et al.* (2012) (based on (Hovland *et al.* 1996)) is a model using hidden Markov networks. This stochastic network is built here by learning data from robotic perception systems. The goal is to determine intent using past observations. This model uses the theory of mind by invoking that humans infer the intentions of their peers by using a projection of their own.

Another contextual approach is the one developed for robotics by Hofmann and Williams (2007). The stochastic system is completed by a weighting based on an analysis of vernacular corpuses. We can therefore use the context of an observation to determine the most credible actions using the relational system built with corpus

analysis. This is based on the observation of the objects in the scene and their condition. This makes common sense actions much more likely and almost impossible actions leading to semantic contradictions.

#### 7.2.2.6 Bayesian Theory of Mind

This principle is also used as the basis of the paradigm of Baker and Tenenbaum (2014) which forms a Bayesian theory of the mind. Using a limited representation of the human mind, this model defines formulas for updating beliefs and probabilities a posteriori of world states and actions. This is constructed with sigmoid distributions on the simplex of inferred beliefs. Then the probabilities of desire are calculated in order to recover the most probable intention. This has been validated as being close to the assessment of human candidates on simple intention recognition scenarios.

## 7.3 Inverted planning

doES

Another way to do intent recognition do not rely on having a plan library at all by using inverted planning. In fact, intent recognition is the opposite problem as planning. In planning we compute the plan from the goal and in intent recognition we seek the goal from the plan. This means that planning is a *deduction* problem while intent recognition is an *abduction* problem. It is therefore possible to transform an intent recognition problem into a planning one.

More intuitively, this transformation relies on the *theory of mind*. This notion of psychology states that one of the easiest ways to predict the mind of another agent is by projecting our own way of thinking onto the target. The familiar way to understand this is to ask the question, "what would **I** do if I were them ?". This is obviously imperfect since we don't have the complete knowledge of the other mind but is often good enough at basic predictions.

This theory is based on the Belief, Desire and Intention (BDI) model. In our case the belief part is akin to the knowledge database, the desires are the set of possible goals (weighted by costs) and the intent is the plan that achieve a selected goal.

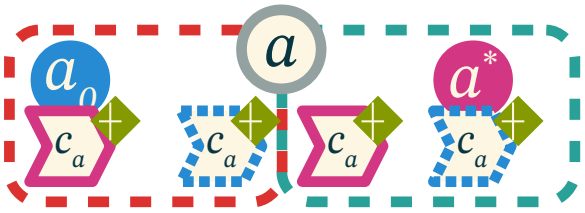
A good analysis of this way of thinking in the context of intent recognition can be found in **CITATION** (Baker)'s work on the subject.

To get further, the work of **CITATION** (Ramirez) is the founding paper on the principle of transforming the intent recognition problem into a planning one. That work was later improved by **CITATION** (Chen) in order to support multiple and concurrent goals at once.

In order to do that Ramirez rediscovers an old tool called constraints encoding into planning **CITATION**. This allows to force the selection of operators in a given a certain order or adding arbitrary constraints on the solution. ???

pas de majuscule

The encoding on itself is quite straight forward : Adding artificial fluents to derive an action's behavior considering its selection. In the case of Ramirez's work, the fluents




citer/expliquer a figure ....

Figure 7.4: Representation of the encoding of constraints using extra fluents.

ensure that the observed actions are selected at the start. The resulting plan is then compared to another plan computed while avoiding the observed action. The difference in cost is proportional to the likelihood of the goal to be pursued.

This problem transformation was more recently improved significantly by **CITATION** (Sohrabi). Indeed, their work allows for using observed fluents instead of actions. This modification allows for a more accurate and flexible prediction with less advanced observations. It also takes into account the missing and noisy observation to affect negatively the likelihood of a goal. Along with the use of diverse planning, this technique allows for seamless multi-goal recognition.

In order to see this technique used in practice we refer to the works of **CITATION** (Tamadupula).

### 7.3.1 Probabilities and approximations

c'est toi qui propose cette preuve ? ou tu reprends la littérature ? préciser

Now that the intuition is covered, we need to prove that the result of the planning process is indeed correlated to the probability of the agent pursuing that plan knowing the observations. But first we need to formalize how probabilities work.

An *event* is a fixed fluent, a logical proposition that can occur. The likelihood of an event happening ranges from $0$ (impossible) to $1$ (certain). This is represented by a relation named **probability** of any event $e$ noted $\mathbb{P}(e) \in [0,1]_{\mathbb{R}}$. This means that probabilities are real numbers restricted between $0$ and $1$.

**Definition 53** (Conditional probabilities)**.** Conditional probabilities are probabilities of an event assuming that another related event happened. This allows to evaluate the ways in which events are affecting one another. The probability of the event $A$ occurring knowing that $B$ occurred is written:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

with, $\mathbb{P}(A \cap B)$ being the probability that both events occur. In the case of two *independent* events $\mathbb{P}(A|B) = \mathbb{P}(A)$.

We note the set of goals that can be pursued $G$ and the temporal sequence of observations $\mathcal{O}$. Using conditional probabilities, we seek to have a measure of $\mathbb{P}(g|\mathcal{O})$ for any goal $g \in G$.

In that section we explain how inverted planning does that computation.

For any set of observations $\mathcal{O}$ the probability of the set is the product of the probability of any observation $o \in \mathcal{O}$. We can then note $\mathbb{P}(\mathcal{O}) = \prod_{o \in \mathcal{O}} \mathbb{P}(o)$.

We assume that the observed agent is pursuing one of the known goals. The event of an agent pursuing a specific goal is noted $g$. This means that $\mathbb{P}(G) = \sum_{g \in G} \mathbb{P}(G) = 1$ because the event is considered certain.

Using *conditional probabilities* we can also note $\mathbb{P}(G|\pi) = 1$ for a valid plan $\pi$ that achieves any goals $g \in G$.

**Theorem 4** (Bayes). *Bayes's theorem allows to find the probability of an event based on prior knowledge of other factors related to said event. It is another basic way to compute conditional probabilities as follows:*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

In the Bayesian logic, one should start with a *prior* probability of an event and actualize it with any new information to make it more precise. In our case, we suppose that $\mathbb{P}(G)$ is given or computed by an external tool.

From the direct application of Bayes's theorem and the previous assumptions, we have :

$$\mathbb{P}(\pi|\mathcal{O}) = \frac{\mathbb{P}(\mathcal{O}|\pi)\mathbb{P}(\pi)}{\mathbb{P}(\mathcal{O})} = \frac{\mathbb{P}(\mathcal{O}|\pi)\mathbb{P}(\pi|G)\mathbb{P}(G)}{\mathbb{P}(\mathcal{O})}$$

{#eq:plan-obs}

Using the event $\pi$ transitively we can simplify to:

$$\mathbb{P}(G|\mathcal{O}) = \frac{\mathbb{P}(\mathcal{O}|G)\mathbb{P}(G)}{\mathbb{P}(\mathcal{O})}$$

{#eq:goal-obs}

**Theorem 5** (Total Probability). *If we have a countable set of disjoint events $E$ we can compute the probability of all events happening as the sum of the probabilities of each event:*

$$\mathbb{P}(E) = \sum_{e \in E} \mathbb{P}(e)$$

Since we consider that we have all likely plans for a given goal we can neglect the very improbable ones and assert that the events of any given plans being acted are independent from one another. Also, the total probability of all plans is certain. From the total probability formula:

$$\mathbb{P}(\mathcal{O}|G) = \sum_{\pi \in \Pi_G} \mathbb{P}(\mathcal{O}|\pi)\mathbb{P}(\pi|G)$$

{#eq:obs-goal}

In equation **??**, we have $\mathbb{P}(G)$ and $\mathbb{P}(\mathcal{O})$ known from prior knowledge. Along with equation **??**, we can say that:

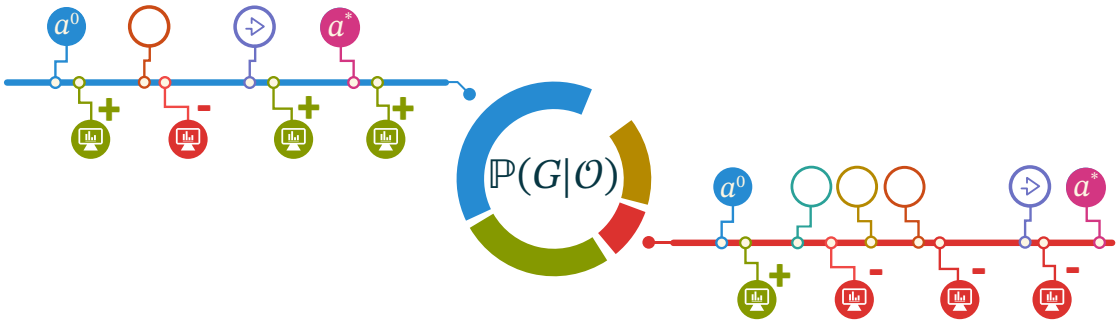
$$\mathbb{P}(G|\mathcal{O}) = \alpha\mathbb{P}(\mathcal{O}|G)\mathbb{P}(G)$$

With $\alpha$ being a normalizing constant. Also, using the previous formula we can assert that:

$$\mathbb{P}(g|\mathcal{O}) \propto \sum_{\pi \in \Pi_g} \mathbb{P}(\pi|\mathcal{O})$$

This means that if the cost of the plan is related to its likeliness of being pursued knowing the observation sequence, we can evaluate the probability of any goal being pursued. This allows for Sohrabi's problem transformation to work.

That transformation is simply affecting the cost of a plan by dissuading any missed or added fluents while rewarding correct predicted fluents relative to the observations. This process is illustrated in figure 7.5.




expliquer la figure on ne comprends rien !

Figure 7.5: Illustration of how plan costs are affecting the resulting probabilities.

## 7.4 Intent recognition Using Abstract Plans

The initial objective of this thesis was to make intent predictions use abstract plans and repairs. Since plan repair is very susceptible to the heuristic, it is not a reliable tool for general and uncertain planning and will perform worse on all cases not handled well by a given heuristic.

This is not the case of abstract HTN planning since the plan generated is valid while incomplete. The idea here is to quantify the quality of an abstract plan to weigh its plausibility and the likelihood of missing fluents using Sohrabi's method.

The unforeseen problem is that, while in theory this seems like a very efficient approach, in practice it can lack a lot of performance since turning a sequence of observed fluents into a backward chaining heuristic is tricky at best.

In this section, some idea of how that could be done is explored along with perspectives for further works regarding the subject.

### 7.4.1 Linearization

In order to use the approach of Sohrabi, we need total ordered plans. This is quite an issue since our planner generates partial order plans. Each of these plans have one or several *linearizations*: totally ordered plans that correspond to all possible orders of the plan.

The idea here is to merge parallel actions into one using graph quotient. This is the same mechanism behind sheaves. To do this we use the fact that there are no threats in valid plans and therefore parallel actions have compatible preconditions and effects. This allows to merge several actions into one that is equivalent in terms of fluents and cost. To merge two actions into one we do the following: $\mathit{pre}(a_m) = \mathit{pre}(a_1) \circ \mathit{pre}(a_2)$ and $\mathit{eff}(a_m) = \mathit{eff}(a_1) \circ \mathit{eff}(a_2)$. We use the application of states over other states and ignore the order.

### 7.4.2 Abstraction

Since abstract plans use composite actions that have explicit preconditions and effects, they can be treated as a normal action. Indeed, while it is possible that an abstract action have less requirements and effects than what is done in its methods, it can't have more since the methods must *at least* fullfil the parent action's "contract".

Adding to that, while merging actions in the linearization step, it may be appropriate to merge actions into a composite action that holds all the merged actions in its only method.

### 7.4.3 Example

In this section we present an example as well as a description of the execution of our planning algorithm and how the results are used for intent recognition.

**Example 53.** Figure 7.6 illustrates an example of such a linearization. The linearization happens by merging actions that can be done simultaneously. Of course, this is not a valid totally ordered plan since classical planning supposes that the agent can only perform an action at once.

In our example, a person wants to dress themselves. In a partial order plan there is no need to order between top clothing and bottom clothing. It is also unecessary to specify if the right or left socks and shoes should be put on first.

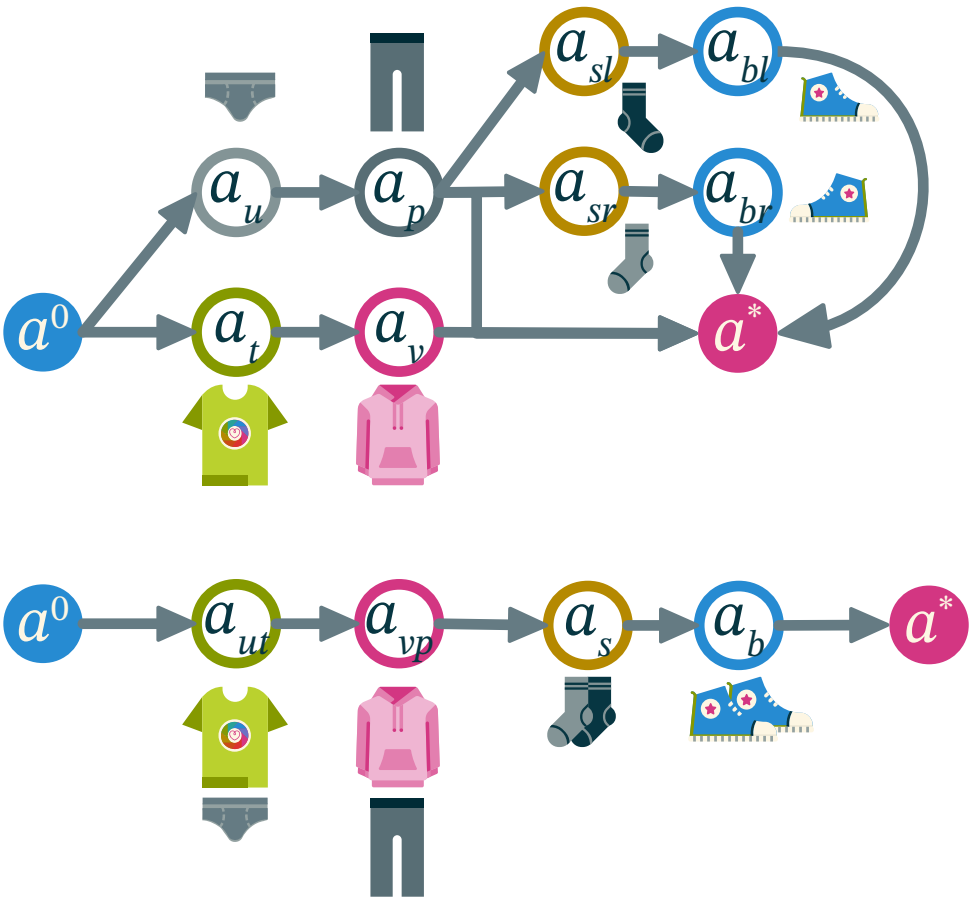



Figure 7.6: Example of linearization of partial order plan.

In a first time, we explain how the planning algorithm can affect the result of the intent recognition.

#### 7.4.3.1 Partial Order Approach

Using a previously described POP algorithm (chapter 6) we can create a planning domain with the following actions from the example in figure 7.6: $a_u, a_t, a_p, a_v, a_{s(l|r)}$ and $a_{b(l|r)}$. Each action intinctively requires you to wear clothes that are underneath their related garmant and put it on. For example the action $a_{br}$ requires the right socket to be put on (effect of $a_{sr}$) and will result in putting the right shoe on.

The goal is to put on every clothes from a state of none are already put on. The planner will select each action to fulfil the goal and add causal links to prevent threats between actions and to fulfil their preconditions. The resulting plan is the upper one in figure 7.6.

Now the intent recognition part quicks in and linearize the plan. To do so, the algorithm identifies actions that can be done simultaneously and merge them into a single action. The purpose of this merging is to make the plan totally ordered while keeping fluents in chronological order. Here we can see that the linear plan describes actions of puting clothe by layers instead of by garment. Details are lost but the chronological order and cost is kept (by adding costs of all actions being merged).

This allows for existing intent recognition methods to be applied on the resulting plan without loosing on the advantages of partial order plans. This technique results in the plan on the bottom of figure 7.6. This plan is totally ordered and can be used by the classical inverted planning approaches as described previously.

#### 7.4.3.2 Abstract plan Approach

This example also illustrates the advantages of using abstract plans for intent recognition. If the domain is properly designed for it, we can use an abstract plan that has composite actions made by layers of clothing.

In that approach, we re-use the previous domain but extend it with the following actions: $a_{ut}, a_{vp}, a_s, a_b$. These actions will behave in the same way as the merged actions of the linearization. Each action correspond to a layer of clothing and can be decomposed into the atomic actions relative to that layer. For example, the action $a_{ut}$ is the action related to putting on the first layer of clothing. It can be decomposed into the action of putting underwear $a_u$ and the action of putting on a t-shirt $a_t$.

Using our HEART planner, it is possible to request only an abstract plan containing this level of actions. The planning process will therefore result into the plan on the bottom of figure 7.6 that is also the linearized plan found earlier.

Since the abstract plans are easier to compute, the intent recognition becomes faster for the same result in that case. The main factor in the efficiency of this method is the design of the domain.

## 7.5 Conclusion

In this chapter, we present existing intent recognition techniques and expose how inverted planning can be fitted to our planning approach.

# 8 Conclusion and Perspectives

Et la conclusion avant les perspectives ? Tu dois conclure ta thèse, en rappelant tout ce que tu as présentés et proposé (cela permettre peut être au lecteur de mieux comprendre tes réelles contributions qui restent un peu noyées dans le reste dans ton rapport) et validé

## 8.1 Perspectives and discussions

???

<!–> *"the discipline underlying ZF will be the **first-order predicate calculus**. The primitive symbols of this set theory, taken from logic, are the connectives, quantifiers and variables mentioned above and, possibly, also the symbol of equality (discussed below), as well as such auxiliary symbols as commas, parentheses and brackets."*–>

Fraenkel *et al.* (1973, 67,22)

### 8.1.1 Knowledge representation

Listing the contributions there are a couple that didn't make the cut. It is mainly ideas or projects that were too long to check or implement and needed more time to complete. SELF is still a prototype, and even if the implementation seemed to perform well on a simple example, no benchmarks have been done on it. It might be good to make a theoretical analysis of OWL compared to SELF along with some benchmark results.

On the theoretical parts there are some works that seems worthy of exposure even if unfinished.

#### 8.1.1.1 Literal definition using Peano's axioms

The only real exceptions to the axioms and criteria are the first statement, the comments and the liberals.

For the first statement, there is yet to find a way to express both inclusion, the equality relation and solution quantifier. If such a convenient expression exists, then the language can become almost entirely self described.

Comments can be seen as a special kind of container. The difficult part is to find a clever way to differentiate them from regular containers and to ignore their content in the regular grammar. It might be possible to at first describe their structure but then they become parseable entities and fail at their purpose.

Lastly, and perhaps the most complicated violation to fix: laterals. It is quite possible to define literals by structure. First we can define boolean logic quite easily in SELF as demonstrated by listing 8.1.

```
1 ~(false) = true;
2 ( false, true ) :: Boolean;
3 true =?; //conflicts with the first statement!
4 *a : ((a | true) = true);
5 *a : ((false | a) = a);
6 *a : ((a & false) = false);
7 *a : ((true & a) = a);
```

Listing 8.1: Possible definition of boolean logic in SELF.

Starting with line 1, we simply define the negation using the exclusive quantifier. From there we define the boolean type as just the two truth values. And now it gets complicated. We could either arbitrarily say that the false literal is always parameters of the exclusion quantifier or that it comes first on either first two statements but that would just violate minimalism even more. We could use the solution quantifier to define truth but that collides with the first statement definition. There doesn't seem to be a good answer for now.

From line 4 going on, we state the definition of the logical operators $\wedge$ and $\vee$. The problem with this is that we either need to make a native property for those operators or the inference to compute boolean logic will be terribly inefficient.

We can use Peano's axioms (1889) to define integers in SELF. The attempt at this definition is presented in listing 8.2.

```
1 0 :: Integer;
2 *n : (++(n) :: Integer);
3 (*m, *n) : ((m=n) : (++m = ++n));
4 *n : (++n ~= 0);
5 *n : ((n + 0) = n);
6 (*n, *m) : ((n + ++m)= ++(n + m));
7 *n : ((n × 0) = 0);
8 (*n, *m) : ((n × ++m) = (n + (n × m)));
```

Listing 8.2: Possible integration of the Peano axioms in SELF.

We got several problems doing so. The symbols * and / are already taken in the default file and so would need replacement or we should use the non-ASCII × and ÷ symbols for multiplication and division. Another more fundamental issue is as previously discussed for booleans: the inference would be excruciatingly slow or we should revert to a kind of parsing similar to what we have already under the hood. The last problem is the definition of digits and bases that would quickly become exceedingly complicated and verbose.

For floating numbers this turns out even worse and complicated and such a description wasn't even attempted for now.

The last part concerns textual laterals. The issue is the same as the one with comments but even worse. We get to interpret the content as literal value and that would necessitate a similar system as we already have and wouldn't improve the minimalist aspect of things much. Also we should define ways to escape characters and also to input escape sequences that are often needed in such case. And since SELF isn't meant for programming that can become very verbose and complex.

#### 8.1.1.2 Advanced Inference

The inference in SELF is very basic. It could be improved a lot more by simply checking the consistency of the database on most aspects. However, such a task seems to be very difficult or very slow. Since that kind of inference is undecidable in SELF, it would be all a research problem just to find a performant inference algorithm.

Another kind of inference is more about convenience. For example, one can erase singlets (containers with a single value) to make the database lighter and easier to maintain and query.

#### 8.1.1.3 Queries

We haven't really discussed quarries in SELF. They can be made using the main syntax and the solution quantifiers but efficiency of such queries is unknown. Making an efficient query engine is a big research project on its own.

For now a very simplified query API exists in the prototype and seems to perform well but further tests are needed to assess its scalability capacities.

–>

–>

–>