ABORT : Abstract Building Operating in Real Time as a refinement strategy for hierarchical partial order planning (Draft)

Antoine Gréa

Introduction

The domain of automated planning is considered a basis of artificial intelligence. As such it must adapt to suit very different needs. Some works might require a fast real time scheduling process to build a path in complex environments. Other works will require flexibility for cooperation or competition with other agents. That is the reason behind the diversity of the planning comunity approach wise.

Our research aims at a totally separated domain : intent recognition and robotics. In the recent years several works extended what is known in psychology as the *theory of mind*. That theory suposes that to recognise other's intents and goals we often use to transpose our own. That is like saying "*what would **I** do ?*" when observing actions of another person. This leads to new ways to use *invert planning* as an inference tool.

One of the first to propose that idea was Baker et al. [1] that uses bayesian planning to infer intentions. Another one was Ramirez et al. [2] that found an elegant way to transform a plan recognition problem into classical planning. This is done simply by encoding observation constraints into the planning domain [3] to ensure the selection of actions in the order that they were observed. A cost comparison will then give us a probability of the goal being pursued given the observations. Some works extended this with multi-goal recognition [4] and robotic applications [5].

Very recently, another paper extended this approach significantly. The method proposed by Sohrabi et al. [6] make the recognition at the fluent level instead of actions. It assign costs to missing or noisy observed fluents by using soft constraints (often called user preferences). This method also uses a meta-goal that combine each possible goal and is realised when at least one of them is reached.

Sohrabi et al. states that the quality of the recognition is directly linked to the quality of the generated plans. This is why guided diverse planning was prefered along with the posibilities to infer several possible goals at once.

Since our applicative domain also include robotics, we wish to account also for the real time aspect of the prediction. Indeed, a prediction is only useful *before* it happens. The human aspect of our context also asks for high level description and extensive domains.

Obviously these work hypothesis are in direct contradiction : the harder the problem, the longer the resolution. And this is especially true in automated planning as it has been proven to be a P-SPACE problem if not harder. A big and complex domain is intractable on limited embeded robotics.

Our interest was on the following question : what happens when one *can't* find the solution to the problem ? What if the problem can't be solved ? What if we ran out of time ? This question has been partially threated in [7] where "excuses" are being investigated as response to unsolvability.

Figure 1: A potential logo.

SPECULATION : In our approach we show that abstract solutions are often enough if not better suited for human interaction. The key is in better domain description and use. Using Hierarchical Task Networks (HTN) and Partial Order Planning (POP) we can build iteratively better and more complete plans while being able to provide abstract plans whenever it is needed. The last interest of this work is in perspective shifting or how a robot can resume a job started by a human with different capabilities.

Table 1: Most used symbols in the paper. **TODO: Update that**

Symbol	Description
$pre(o), eff(o)$	Preconditions and effects of the operator o
$\mathscr{D}$	Planning domain
$\mathscr{P}$	Planning problem
$l_{\rightarrow}, l_{\leftarrow}$	Source and target of the causal link l
$o_1 \succ o_2$	Precedence operator (o_1 precedes o_2)
$\mathscr{F}^{\pm}(\pi)$	Set of flaws in π
$r(f)$	Resolvers of the flaw f
$f.n$	Needer of the flaw f
$f(\pi)$	Application of the flaw f on plan π
$\Downarrow \pi$	Full support of π
$\pi \models \mathscr{P}$	The partial plan π is a valid solution of $\mathscr{P}$

1 Definitions

We aim to define all necessary notions in order to make a framework in which we build our planner. This framework is based on reusability and recursive notions. Such properties make further definition simpler and factorize the work as we only need to define future notions for a reduced set of cases.

1.1 Notations

In this paper, we use the notation defined in table 1. Our notation is adapted from the one used in [8].We use the symbol $\pm$ to signify that there is a notation for the positive and negative symbols but the current formula works regardless of the sign. All related notions will be defined later.

1.2 Logic description

Before the planning specific definition, we must introduce the elements of our fluent representation. To simplify, it is first order logic mixed into RDF representation.

Definition 1 (Entity)**.** An entity e is an abstract object with a few attributes :

- a **name** that may be empty,
- a **type** that caracterize its nature.

Some entities have additional attributes in function of their type :

- **parameters** that is simply an ordered group of entities,
- a **value** that is intrisic to the given entity and that can eventually change.

In our representation, contrarily to standard RDF, everything is entity : types are entities, groups are entities, statements are entities, the language itself is an entity.

That means that we can represent *metastatements* a.k.a. statements about statements.

In this context, a fluent is a statement about the world. It can use parameters (in the scope of the current or host statement) and enventually quantifiers.

TODO Reformulate this

- T is the set of **types** used in the domain for variables and constants. We provide types with a relation of **subsumption** noted $t_1 \prec t_2$ with $t_1, t_2 \in T$ meaning that all instances of t_1 are also instances of t_2.
- $C_{\mathscr{D}}$ is the set of typed **domain specific constants**.
- R is the set of **relations** (also called properties). They are used to express aspects and properties of the world. We note them as a tuple $r = \langle name, r_s, r_t \rangle$ with $name$ as the relation's name, r_s and r_t as the source and target type and arity (ex.: $Object^2$ signify a pair of objects). When a relation is instanciated it is called a *fluent* and binds an object to n subjet(s) (n being the arity of the relation).
- F is the set of possible **fluents**. An additional unary relation is needed on the fluents : $\neg f$ is used to signify that the fluent is negative. We also add special behavior to sets of fluents. For exemple we note $F^{\pm}$ the set of positive/negative fluent present in F. A set of fluents can be either seen as a *state*, *effects* or *preconditions* :
  - *Preconditions* have a predicate of validity noted $s \models p : \left[(p^+ \in s) \wedge (p^- \notin s) \right]$, meaning that all positive fluents must be present and all negative fluents absent in the state s for the precondition p to be valid.
  - *Effects* have an application operation noted $s + e = (s \setminus e^-) \cup e^+$. The values erasive, meaning that if a fluent is present with different target value the value in the state s is erased by the value in e except if the target arrity of the relation of the fluent is infinite. **TODO: example to explain maybe + open/closed world hypothesis**

1.3 Domain definitions

Definition 2 (Domain)**.** Our domain definition takes into acount some modifications necessary to achieve the type of planning we aim for. A domain is a tuple $\mathcal{D} = \langle T, C_{\mathcal{D}}, R, F, O \rangle$ where

- F is the set of **fluents** as described earlier. **TODO action in their dedicated section**
- O is the set of **operators**. An operator is a tuple $o = \langle name, pre, eff, methods, lv \rangle$ with $name$ as the operator's name, pre as the set of fluent coresponding to a conjonction that must be met to apply the operator using the set of fluent that represents the effect : eff. We note $methods$ is the set of *methods* of the operator that contains a partial plan to realise the operator. The set can be empty in witch case the operator is said to be *atomic*. l is the abstraction level of the operator.

This definition lays some important bases of our framework. Our operators are **composite** meaning that a *method* must be chosen and instanciated as a partial plan in order to complete any plan containing an instanciation of this operator.

We note $lv_{max}(\mathcal{D})$ the maximum abstraction level of a given domain.

1.4 Partial plan definitions

Definition 3 (Partial Plan)**.** We define a partial plan as a tuple $\langle S, L, B \rangle$ with :

- S the set of **steps**. A step is a partially or fully instantiated operators also called *actions*. It has two unary relations :
  - $lv_i(a)$ whitch affects to each step an interval of integers that corresponds to the **abstraction levels of the step**.
  - $dummy(a)$ is a predicate that gives true iff the step is a **dummy action**. Dummy actions are always linked to their parent via a *hierarchical causal link* and cannot be removed without removing the parent action.
- L the set of **causal links**.
- B the set of **binding constraints**.

Partial plans can be seen as a simple form of planning graphs. It is a directed acyclic graph with action as vertices and causal links as edges. In the case of HTN we add another dimensionality to the planning graph with *abstraction levels*. On the highest level we use the most generic composite operators. The last level is for atomic actions belonging to the most specific composite operators. Each level can be seen as a plan by itself. We introduce the set of **hierarchical causal links** that is a subset of $L_H \subset L$. These links are not to be acounted for support (see definition 4). Another propriety of these levels is that $lv(o) <= lv_i(a)$ with a being an instanciation of o. We can also extract regular partial plans from leveled partial plans at any level. π_n is a restriction of the leveled partial plan π at level n.

We note $l : a_s \xrightarrow{c} a_t$ the **causal link** l that binds the *source step* a_s to the *target step* a_t using the *cause* c. A cause is a simple set of fluents that is provided by the source to the target.

Definition 4 (Support)**.** We define support as a property of preconditions, steps and partial plans. A partial plan π is fully supported if each of its steps $o \in S$ is fully supported. A step is fully supported if each of its preconditions $f \in pre(o)$ is supported. A precondition is fully supported if there exists a causal link l that provides it. We note full support of a partial plan as :

$$\pi \Downarrow \equiv \begin{array}{l} \forall a \in S \ \forall f \in pre(a) \ \exists l \in L^-_\pi(a) \setminus L_H : \\ (f \in l \land \nexists t \in S(l_\rightarrow \succ t \succ a \land \neg f \in eff(t))) \end{array}$$

with $L^-_\pi(a)$ being the incoming causal links of a in π and $l_\rightarrow$ being the source of the link.

This means that in order to be fully supported a partial plan needs to have all its step's precondition supplied with causal links without possible order that allows a step to threaten another link. We add the notation $\pi \downarrow$ for partial support, meaning that some preconditions might not be supported. Obviously preconditions cannot be partially supported.

A specificity of our definition is that the ordering constraints are part of causal links. When an ordering constraint doesn't have a cause we use a **bare causal links** (mostly used for threats). We also introduce the **precedence operator** noted $a_i \succ a_j$ with $a_i, a_j \in S$ iff there is a path of causal links that connects a_i to a_j. We call a_i *anterior* to a_j.

The notion of support can be extended to a single level $\pi \Downarrow^n$

1.5 We have a problem

Definition 5 (Problem)**.** The planning problem is defined as a tuple $\mathcal{P} = \langle \mathcal{D}, C_{\mathcal{P}}, \Omega \rangle$, where

- $\mathcal{D}$ is a planning domain,
- $C_{\mathcal{P}}$ is the set of **problem constants** disjoint from the domain constants,
- Ω is the **root operator** which methods correspond to the solution of the problem. Its level is set to $lv_{max}(\mathcal{D}) + 1$.

The root operator contains the **initial state and goal** (noted respectively I and G) of the problem. In the context of multiple planning, this operator may have several methods that corresponds to several result plans.

1.6 Plan refinement definitions

Partial Order Planning fixes flaws in a partial plan to refine it into a valid plan that is a solution to the given problem. Since we work with a HTN approach the classical flaws must be adapted [9]. Therefore, we present the updated flaws for our version of POP.
Definition 6 (Subgoal)**.** A flaw in a partial plan, called subgoal, is a missing causal link required to support a precondition of a step. We can note a subgoal as $a_n \not\Downarrow_s$ with s being **proper fluent** of the subgoal (also called *open condition*) and also a unsuported precondition of the action a_n called the **needer** of the subgoal.

It is to be noted that hierarchical causal links does not account for support and therefore doesn't solve subgoals. Also flaws can only be applied at a given abstraction level.
Definition 7 (Threat)**.** A flaw in a partial plan called threat consists of having an effect of a step that can be inserted between two actions with a causal link that is threatened by the said effect. We say that a step a_b is threatening a causal link $a_p \xrightarrow{t} a_n$ iff $a_b \neq a_p \neq a_n \wedge \neg t \in eff(a_b) \wedge a_p \succ a_b \succ a_n$ with a_b being the **breaker**, a_n the *needer* and a_p a *provider* of the *proper fluent* t.

Flaws are fixed via the application of a resolver to the plan. A flaw can have several resolvers that match its needs.
Definition 8 (Resolvers)**.** A resolver is a potential causal link defined as a tuple $r = \langle a_s, a_t, f \rangle$ with:

- $a_s, a_t \in S$ being the source and the target of the resolver,
- f being the considered fluent.

For classical flaws, the resolvers are simple to find. For a *subgoal* the resolvers are sets of the potential causal links between a possible provider (along allowed operators or steps) of the proper fluent and the needer. To solve a *threat* there are mainly two resolvers: a causal link between the needer and the breaker called **demotion** or a causal link between the breaker and the provider called **promotion**.

Once the resolver is applied, the algorithm needs to take into account **side effects** of that application.
Definition 9 (Side effects)**.** Flaws that arise because of the application of a resolver on the partial plan are called causal side effects or *related flaws*.

We can derive this definition for subgoals and threats:

- **Related Subgoals** are all the new open conditions inserted by new steps.
- **Related Threats** are the causal links threatened by the insertion of a new step or deletion of a causal link.

2 Algorithm

3 Properties of ABORT

4 Results

Conclusions

References

[1] C. L. Baker, J. B. Tenenbaum, and R. R. Saxe, "Goal inference as inverse planning," in *Proceedings of the 29th annual meeting of the cognitive science society*, 2007.

[2] M. Ramirez and H. Geffner, "Plan recognition as planning," in *IJCAI*, 2009, pp. 1778–1783.

[3] M. Baioletti, S. Marcugini, and A. Milani, "Encoding planning constraints into partial order planning domains," in *PRIN-*

CIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-, 1998, pp. 608–616.

[4] J. Chen, Y. Chen, Y. Xu, R. Huang, and Z. Chen, "A Planning Approach to the Recognition of Multiple Goals," *International Journal of Intelligent Systems*, vol. 28, no. 3, pp. 203–216, 2013.

[5] K. Talamadupula, G. Briggs, T. Chakraborti, M. Scheutz, and S. Kambhampati, "Coordination in human-robot teams using mental modeling and plan recognition," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014, pp. 2957–2962.

[6] S. Sohrabi, A. V. Riabov, and O. Udrea, "Plan Recognition as Planning Revisited," 2016.

[7] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming Up With Good Excuses: What to do When no Plan Can be Found." in *ICAPS*, 2010, pp. 81–88.

[8] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: Theory & practice*. Elsevier, 2004.

[9] P. Bechon, M. Barbier, G. Infantes, C. Lesire, and V. Vidal, "HiPOP: Hierarchical Partial-Order Planning," in *STAIRS 2014: Proceedings of the 7th European Starting AI Researcher Symposium*, 2014, vol. 264, p. 51.