Argumentation Driven Planning

Guillermo R. Simari grs@cs.uns.edu.ar

Alejandro J. García ajg@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
Departamento de Ciencias de la Computación
UNIVERSIDAD NACIONAL DEL SUR

BAHIA BLANCA - ARGENTINA

The research line reported here involves developing an argumentation-based formalism that an agent could use in constructing plans. The argumentation formalism will be combined with well known planning tecniques producing a novel way of constructing a plan. Below, we wil sketch the formalism and introduce some clarifying examples.

We assume that the agent has certain knowledge about the world represented by a consistent set of facts Φ and a set of defeasible rules Δ. This knowledge base, represented by $K=(\Phi,\Delta)$, is in fact a restricted Defeasible Logic Program. We will freely use the results already obtained for such argumentation-based extension of logic programming called DeLP [1].

The agent will also have a set of actions Γ that will be used for constructing the plan. Each action *A* will consist of an ordered triplet $\langle \mathbf{X}, \mathbf{P}, \mathbf{C} \rangle$, where **X** is a consistent set of literals representing consequences of executing *A*, **P** is a set of literals representing preconditions for *A*, and **C** is a set of constrains of the form *not L*, where *L* is a literal. For example, an action for printing a document will require a connected and working printer, the file to be printed, and the constrain that the printer is not out of paper. After executing the action the consequence will be to have the printed document. Suppose that *f* is a file and *lp* is a printer, then this action could be represented by

$$\langle \{\text{printed\_file}(f)\}, \{\text{connected}(lp), \text{working}(lp)\}, \{\text{not out\_of\_paper}(lp)\} \rangle$$

For executing an action *A*, the action must be applicable, that is, the preconditions **P** of *A* must hold and the constrains **C** of *A* must not hold. Usually the knowledge that an agent has about the current state of the world is represented by a consistent set Φ of literals known to be true. Therefore checking if a literal *L* holds is simply checking if *L* is in the set Φ.

However, as mentioned before, here the knowledge of an agent will be represented by a restricted defeasible logic program (Φ,Δ). Using this approach, a literal *h* will hold when *h* is warranted. In DeLP a literal *h* is warranted if the argument **A** that supports *h* has no *defeaters*, or every defeater for **A** is defeated (see [1] for details).

Therefore, given an action $A = \langle \mathbf{X}, \mathbf{P}, \mathbf{C}\rangle = \langle\{P_1,\ldots, P_m\}, \{X_1,\ldots, X_n\}, \{\text{not } C_1,\ldots,\text{not } C_k\}\rangle$ in Γ, the action *A* will be applicable if every precondition P_i in **P** has a *warrant* built from (Φ,Δ), and every constraint C_i in **C** fails to be warranted.

The effect of executing an action $A = \langle \mathbf{X}, \mathbf{P}, \mathbf{C}\rangle$, will be the revision of Φ by **X**, i.e.

$$\Phi^{*\mathbf{X}} = \Phi^{*\{X_1,\ldots,X_n\}}.$$

Revision will consist in removing any literal in Φ that is complementary of any literal in **X** and then adding **X** to the resulting set. Formally: $\Phi^{*\mathbf{X}} = \Phi^{*\{X_1,\ldots,X_n\}} = (\Phi - \overline{\mathbf{X}}) \cup \mathbf{X}$ where $\overline{\mathbf{X}}$ represents the set of complements of members of **X**. Hence, after executing an applicable action, the set Φ will change.

Assume that the agent's knowledge base $K = (\Phi,\Delta)$ has the set of facts

$$\Phi = \{ \text{cast\_away(h), at(h,beach), is(raining), has(h,coconut) } \}$$

expressing that h is a cast away, h is at the beach, and it is raining.

And the set Δ of defeasible has the rule "has(X,sharp_stone) $\longrightarrow$ at(X,beach)" expresing that *"usually if an agent is at the beach, it has a sharp stone''*

Suppose that the agent has the following action for making a container:

make_container =

$$\langle\{\text{has(h,container), ~has(h,coconut)}\}, \{\text{has(h,sharp\_stone), has(h,coconut)}\}, \{\}\rangle$$

The action has one precondition and no constrains. Observe that the precondition ``has(h,sharp_stone)'' is not in the set $\Phi$, however, there is a rule in $\Delta$ that allows us to build an argument for ``has(h,sharp_stone)'' that has no defeaters, so it becomes warranted. Therefore the action make_container is applicable and when executed it will add to the set Φ the literal "has(h,container)'' and it will remove the literal "has(h,coconut)".

If one action *A* is not applicable because a precondition *P* is not warranted, then the agent could search for other applicable action *B* that could change the state of the world adding *P* to the set Φ, executing *B*, and after that executing *A*. Thus, action will be chained. For example,

collect_rain = ⟨{has(h,water)}, { has(h,container), is(raining))}, {not asleep(h)}⟩

is not applicable in $K = (\Phi,\Delta)$ because there is not warrant for "has(h,container)". However, if the action "make_container" is executed first, then the set Φ is revised and changed to

Φ_1 = { cast_away(h), at(h,beach), is(raining), has(h,container) }

Now, with Φ_1 the action collect_rain is applicable. If collect_rain is executed the set Φ_1 is revised obtaining

Φ_2 = { cast_away(h), at(h,beach), is(raining), has(h,container), has(h,water) }

An action *A* could also fail to be applicable if one of its constraints is warrant. Using the same idea as before, executing other actions previously could produce a change in the world that invalidates the warrant of the constrain, and allowing *A* to be applicable.

Therefore, in order to satisfy certain goals, an agent will change its knowledge Φ performing applicable actions. If a required action *A* is not applicable, other actions can be used for allowing *A* to be applicable. Thus, a sequence of actions (plan) will be obtained.

References

[1] García, A. J. *Defeasible logic programming: Language Definition, Operational Semantics and Parallelism.* Ph.D. thesis, Dep. de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, December 2000.