Supporting Material for IJCAI 2018 Submission

Paper ID: 3999

1 Architecture Description

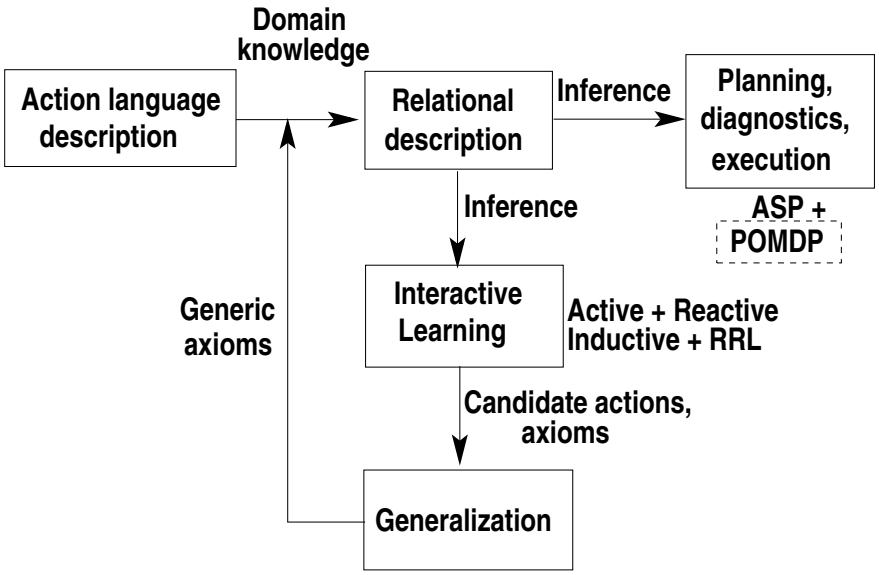

Figure 1: Architecture combines complementary strengths of declarative programming and interactive learning for reasoning with and interactively learning domain knowledge.

Figure 1 presents an overview of the overall architecture. As described at the beginning of Section 2 in the paper, incomplete domain knowledge is encoded in an action language and used to construct tightly-coupled relational representations at two resolutions. For any given goal, each abstract action in a plan computed by non-monotonic logical reasoning at the coarse resolution is executed as a sequence of concrete actions computed by a partially observable Markov decision process (POMDP) that reasons probabilistically over the relevant part of the fine-resolution representation, with action outcomes and observations updating the coarse-resolution history. As stated earlier, we abstract away the reasoning at different resolutions and the probabilistic modeling of perceptual uncertainty, and focus on the interplay between representation, reasoning, and learning. The relational representation is thus translated into an ASP program for planning and diagnostics. ASP-based reasoning also guides the interactive learning of actions, affordances, and the preconditions and effects of actions, using observations of active exploration, reactive execution, and human descriptions—the learned knowledge is then used for subsequent reasoning.

2 Relevant System Description

In the context of learning axioms for different kinds of knowledge in Section 2.2 ("Interactive Learning"), the paper states that ASP-based reasoning automatically restricts the object constants, domain attributes and

axioms *relevant* to a transition being explored. This notion of relevance is based on the following observations regarding the relations that may appear in a discovered axiom:

- For any static attribute that may exist in the body of the discovered axiom, we wish to explore all possible elements in the range of the attribute, e.g., for action $serve(rob_1, cup_1, person_1)$, all possible weights of cup_1 and roles of $person_1$ are explored.
- For any fluent that may appear in the body of the axiom, we wish to explore only those elements in the range of the fluent that occur in the state before or after the state transition. Any other element cannot, by design, be influenced by this transition anyway.

ASP-based reasoning is used to encode these assumptions and automatically construct system description $\mathcal{D}(T)$, the part of $\mathcal{D}$ relevant to the transition T. To do so, we first define the object constants relevant to the transition of interest.

Definition 1 *[Relevant object constants]*
Let a_{tg} be the *target action* that when executed in state σ_1 did not result in the expected transition $T = \langle \sigma_1, a_{tg}, \sigma_2 \rangle$. Let $relCon(T)$ be the set of object constants of Σ of $\mathcal{D}$ identified using the following rules:

1. Object constants from a_{tg} are in $relCon(T)$;
2. If $f(x_1, \ldots, x_n, y)$ is a literal formed of a domain attribute, and the literal belongs to σ_1 or σ_2, but not both, then $x_1, \ldots, x_n, y$ are in $relCon(T)$;
3. If the body B of an axiom of a_{tg} contains an occurrence of $f(x_1, \ldots, x_n, Y)$, a term whose domain is ground, and $f(x_1, \ldots, x_n, y) \in \sigma_1$, then $x_1, \ldots, x_n, y$ are in $relCon(T)$.

Constants from $relCon(T)$ are said to be *relevant* to T, e.g., for action $a_{tg} = serve(rob_1, cup_1, person_1)$ in the RA domain, with $loc(rob_1, office)$, $loc(cup_1, office)$, and $loc(person_1, office)$ in σ_1, the relevant object constants include rob_1 of sort *robot*, cup_1 and $person_1$ of sort *thing*, and $office$ of sort *place*.

Definition 2 *[Relevant system description]*
The system description relevant to the desired transition $T = \langle \sigma_1, a_{tg}, \sigma_2 \rangle$, i.e., $\mathcal{D}(T)$, is defined by signature $\Sigma(T)$ and axioms. $\Sigma(T)$ is constructed as follows:

1. Basic sorts of Σ that produce a non-empty intersection with $relCon(T)$ are in $\Sigma(T)$.
2. For basic sorts of $\Sigma(T)$ that form the range of a static attribute, all object constants are in $\Sigma(T)$.
3. For basic sorts of $\Sigma(T)$ that form the range of a fluent, or the domain of a fluent or a static, the object constants that are in $relCon(T)$ are in $\Sigma(T)$.
4. Domain attributes restricted to basic sorts of $\Sigma(T)$ are in $\Sigma(T)$.

Axioms of $\mathcal{D}(T)$ are those of $\mathcal{D}$ restricted to $\Sigma(T)$. For $a_{tg} = serve(rob_1, cup_1, person_1)$ in our current example, $\mathcal{D}(T)$ does not include other robots, cups or people in the domain. It can be shown that for each transition in the transition diagram of the system description $\mathcal{D}$, there is a transition in the transition diagram of $\mathcal{D}(T)$. States of $\mathcal{D}(T)$, i.e., literals formed of fluents and statics in the answer sets of the ASP program, are states in the RL formulation, and actions are ground actions of $\mathcal{D}(T)$. Furthermore, it is possible to pre-compute or reuse some of the information used to construct $\mathcal{D}(T)$ for any given T.

These definitions bring up some interesting issues:

1. The extent to which the relevant system description reduces the search space depends on the relationships between the domain attributes and axioms in the domain. For instance, although there are several thousand static attribute combinations and more than a million object configurations in our instantiation of the RA domain, computing the relevant system description often reduces the space of attribute combinations to as few as 12 for the *serve* action. However, in other domains with complex relationships between objects, exploration may need to be further limited to a fraction of this restricted state space.

2. Definitions 1 and 2 are based on some representational assumptions, and may not capture deeper relationships in the construction of the axioms.

3. Finally, Q-learning does not generalize to relationally equivalent states. We address these problems based on the relational representation used in the domain description, as discussed in the paper.