AQL: A Query Language for Action Domains Modelled Using Answer Set Programming

Luke Hopton, Owen Cliffe, Marina De Vos, and Julian Padget

Department of Computer Science
University of Bath, BATH BA2 7AY, UK
lch21@bath.ac.uk, {occ,mdv,jap}@cs.bath.ac.uk

Abstract. We present a new general purpose query and abduction language for reasoning about action domains that allows the processing of simultaneous actions, definition of conditions and reasoning about fluents and actions. AQL provides a simple declarative syntax for the specification of constraints on the histories (the combination of action traces and state transitions) within the modelled domain. Its semantics, provided by the translation of AQL queries into Ans-Prolog, acquires the benefits of the reasoning power provided by Answer Set Programming (ASP). The answer sets obtained from combining the query and the domain description correspond to those histories of the domain changing over time that satisfy the query. The result is a simple, high-level query and constraint language that builds on ASP. Through the synthesis of features it offers a more flexible, versatile and intuitive approach compared to existing languages. Due to the use of ASP, AQL can also be used to reason about partial histories.

1 Introduction

Action domains are a useful mechanism for modelling a variety of domains such as planning, protocol definition and normative frameworks. Given an action description we can use established computational techniques, such as Answer Set Programming (ASP) or SAT solvers to verify or examine model properties. It is desirable that such a system should allow designers to specify model properties with a high degree of flexibility while offering qualitative properties of succinctness and human readability.

Action languages\cite{2,5} are a way of formally describing the effects of actions on a domain using a subset of natural language. Transition systems are central to action languages: with every action (or the combined effects of simultaneous actions) the environment changes. Traditionally action languages have two distinct parts: (i) a description language to capture the effects of actions, thus defining the transition system, and (ii) a query language to write queries or reason about the transition system.

In this paper, we present a new action query and abduction language AQL whose semantics is provided by ASP \cite{1}, a logic programming language used for knowledge representation and reasoning. AQL can be used in two ways: for transition system path selection and for model-checking. AQL extends existing query language to allow for simultaneous actions and the definition of conditions that can then form part of more complex queries. Furthermore, AQL does not rely on the use of any particular action description language but can be used on top of an AnsProlog description or in conjunction with any action language description that maps to AnsProlog.
2 Action Domains

The purpose of action domain \([4, 5]\) modelling is to be able to observe the effect of actions on the environment. A light switch offers a simple example: in one state it is on, the action of flicking a switch changes the state to off, etc. State transitions systems provide an effective representation mechanism, in which each state is identified by a set of fluents that are true in that state. Anything not in the set is not true (for the model). Actions are modelled by transition to a new state, wherein some fluents are added and others removed with respect to the previous state. Here we only consider deterministic domains in which there is exactly one next state resulting from the cumulative effect of a sequence of (sets of) actions—known as a trace—on the current state to build a history: \(s_0 \xrightarrow{a_0} s_1 \xrightarrow{\{a_1, a_2\}} \ldots \xrightarrow{a_m} s_n\), where states are denoted \(s_i, i \in 0 \ldots n\) and actions \(a_j, j \in 0 \ldots m\). Planning \([7]\) is probably the best known example of the use of action domains and reasoning over such domains, where the final state is the goal and the trace is the sequence of actions to achieve that goal.

In this paper, we use answer set programming to represent and reason about action domains, their traces and histories. As a result, histories are output as answer sets. However, rather than modelling domains directly in ASP, we put forward the abstraction of an action language—with subsequent translation to AnsProlog—because this enables the user to focus on domain specifics, rather than generic details—such as inertia—that are common to every domain. Several such action language have been developed, with general languages like A \([5]\), C \([6]\) and DLV-K \([3]\) and numerous domain-specific ones.

3 AQL

AQL is a query language that can be used directly with an AnsProlog program representing the action domain or with any action language.

Given an action domain \(\mathcal{M}\), we use \(\mathcal{A}_\mathcal{M}\) to denote the set of all actions in the action domain \(\mathcal{M}\) while \(\mathcal{F}_\mathcal{M}\) is the set of all available fluents. When modelling histories and traces, we need the monitor the domain over a period of time (or a sequence of states). We assume that they are modelled using \(\text{instant}(I)\). The ordering of instances is established by \(\text{next}(I_1, I_2)\), with the final instance defined as \(\text{final}(I)\). Following convention, we assume that the truth of a fluent \(F \in \mathcal{F}\) at a given state instance \(I\) is represented as \(\text{holdsat}(F, I)\), while an action \(A \in \mathcal{A}\) is modelled as \(\text{occurred}(A, I)\).

AQL has two basic concepts: (i) constraint: an assertion of a property that must be satisfied by a valid trace (e.g. a restriction on which traces are considered), and (ii) condition: a specification of properties that can may hold for a given trace. Conditions can be declared in relation to other conditions and constraints can involve declared conditions. Table \(\ref{tab:AQL-syntax}\) summarises the syntax of the language, while the remainder of this section discusses in detail the elements of the language and their semantics.

**Basic Constructs.** AQL provides (Table \(\ref{tab:AQL-syntax}\)) various forms of names: `variable`, `variable_list`, `name`, `param_list` and `identifier`. An `identifier` is

\(^1\) The action domain will be omitted when it is clear from the context.