State Abstractions, Behavioural Cloning and Reinforcement Learning

Eduardo F. Morales
ITESM-Campus Cuernavaca, Temixco, Morelos, 62589, México

1 Extended abstract

The main idea behind this research is to represent states as sets of relational properties that can be used to characterize a particular state and which may be common to other states. This representation allows us to:

- Create powerful abstractions, as states are characterized by a set of properties rather than physical positions. This makes it useful for large search spaces and applicable to relational domains. In chess, for instance, the state space can be represented by boolean combinations of relational properties between pieces and their positions in the board (e.g., in_check, fork, pin, etc.).

- Learn policies which are, in general, independent of the exact position of the agent and the goal. This allows us to transfer policies to different problems where the same relations apply without any further learning. For instance, in chess the policy that is learned to check mate can be directly applied without any further learning to chess boards of different sizes and to many instances of the goal (i.e., to check mate the opponent in different places starting from different board configurations).

We define an *r-state* as a conjunction of properties represented as first-order relations. The extension of an r-state is the set of states which are covered by its description. Each state is an instance of one and only one *r-state*. An *r-state* can cover a large number of states (i.e., all the states where the relations described in the *r-state* hold). For example, an *r-state* in the chess domain could be *kings_in_opposition and rook_divides_kings*, which covers all the chess positions where these two relations hold (more than 3,000 states). This representation of states effectively creates an abstraction over which the reinforcement learning process is performed.

The set of actions also use a first-order relational representation, similar to STRIPs operators. Rather than trying to apply all the available actions per state, we want to apply only *relevant* actions, i.e., those which apply when particular relations in the state description hold and which, possibly, enforce other relations to hold. An *r-action* is defined by a set of pre-conditions, a *generalized* action, and possibly a set of post-conditions. The pre-conditions are conjunctions of relations that need to hold for the *r-action* to be applicable, and the post-conditions are conjunctions of relations that need to hold after a particular primitive action is performed. The *generalized* action represents all the instantiations of primitive actions which satisfy the conditions. When several primitive actions satisfy the conditions of an *r-action*, one of them is chosen randomly. An *r-action* can be applied to many states, and as expected, not all the *r-actions* apply to all the *r-states*.

We have already applied this abstracted representation based on relational properties on grid problems, the Taxi domain, blocks world problems and a simple chess endgame. In this paper we will first show the main advantages of the approach, particularly in its capability of re-using previously learned policies on similar problems.

The approach assumes that the relations and *r-actions* are pre-defined by the user. In this paper we will show how these relations can be learned using an ILP system and we will illustrated this with examples on chess.

Also, given a set of properties, the *r-actions* can be learned using a behavioural cloning approach. This has the advantage that it defines only a subset of relevant actions per state which substantially simplifies the reinforcement learning process. We have already applied this approach (state abstraction based on relational properties - behavioural cloning - reinforcement learning) to learn how to fly an aircraft (which will be presented at ICML-04). In the paper we will illustrate the advantages of this approach for solving challenging domains with a large number of contiguous variables and noise.

Finally, we will present initial ideas of how to learn both the relations and *r-actions* at the same time and how can this be used within a reinforcement learning framework.