A Proposal for Learning Abstractions used in Angelic Hierarchical Planning

Irvin Hwang

December 22, 2010

1 Overview

Abstraction is one of the most important processes in intelligence because it allows us to reason and achieve goals in complex environments with limited resources. In order to avoid garden paths in understanding abstraction we should study its interaction with problem solving i.e. how abstraction helps us solve problems. Angelic hierarchical planning is a framework for formalizing how abstract actions can be combined to form high-level plans. One issue the framework does not address is how abstract or high-level actions (HLAs) are learned. One way to formalize the notion of abstraction is in terms of least general generalizations, a concept from automatic induction. I propose using techniques for finding least general generalizations to the problem of learning high-level actions.

2 Angelic Hierarchical Planning

Angelic semantics for planning address the problem of how to form abstract plans with high-level actions without reducing these actions to primitives. Suppose I am faced with the problem that I would like orange juice, but I do not have any in my house. A simple plan for solving this problem is to go to the store, buy orange juice, then return home. This is a high-level plan in the sense that each action (e.g. going to the store) can be refined in several different ways into more primitive actions (e.g. find my keys, get into my car, drive to the store or get my bike from the garage, ride my bike to the store). Planning at a high level is crucial because directly searching through sequences of primitive actions is generally not tractable for interesting problems. In order to efficiently plan with high-level actions

we would like to be able to say what their effects are without having to refine them to their primitives. This would allow us to plan in a top-down fashion; for example once I decide my first step is to go to the store I can refine this step without worrying how I will buy juice.

The basic idea is to define the effect (or description) of an action as the set of states it leads to. Now given descriptions for a sequence of actions (a plan) we can determine what the set of reachable states is after the last action is taken. If the set of reachable states for the plan contains the goal state we consider this plan to be a solution for our problem and can proceed to refine the actions in the plan. If the set of reachable states does not contain the goal state we can adjust our plan by considering different (high-level) actions to take; the point being the search space over high-level action sequences is smaller than that of primitive ones. The paper illustrates these ideas with a blocks world-type problem. The exact set of reachable states is often hard to concisely represent, but it is reasonable to believe compact approximations to these sets can be used with this approach as demonstrated in [**?**]. While the ideas presented in the paper are not dependent on any particular representation for state and action, we will assume states, sets of states, actions, and sequences of actions can be represented in a language with variables and function definitions in order to apply least general generalization techniques for learning.

3 Least General Generalization

We use a concept known as least general generalization (lgg) or anti-unification to formalize the notion of abstraction. We would like to be able to think of an object in terms of its structure i.e. as a composition of primitives. A grammar is a compact way to specify how primitives can be combined so one way to capture the structure of an object is to represent it as an expression in some language. An example is the set of natural numbers where 1 is the set of primitives, + is the operator for composition, and the grammar is N -¿ 1 — (+ N N) e.g. 2=(+ 1 1), 4=(+ (+ 1 1) (+ 1 1)), 6 = (+ (+ 1 1) (+ 1 (+ 1 (+ 1 1)))) or (+ (+ 1 (+ 1 1)) (+ 1 (+ 1 1))).

An abstraction of a set can be thought of as the structure shared by elements of the set. Using our expression representation, we can define an abstraction for a set of expressions to be a common subexpression with variables that represent where the expressions differ. An abstraction from the set 2,4,6 might be (+ x x) where x is any natural number. It is worth noting the set defined by the abstraction can be much larger than the set the abstraction came from, e.g. (+ x x) represents the set of all even numbers, but

the original set 2,4,6 is much smaller. A largest common subexpression for a set of expressions is known as a least general generalization (lgg) since the resulting abstraction contains all the expressions it was generalized from, but the lgg is also the most specific abstraction since there is no other abstraction that contains the original set that is also a subset of the lgg. The interesting thing about this notion of abstraction is it may allow us to learn both the high-level actions and their descriptions automatically.

4 Learning High-Level Actions with Least General Generalization

The idea for learning a high-level action is to view an HLA as an abstraction over a set of action sequences. A planning problem consists of a set of states, primitive actions, and a transition function. We can generate successful primitive action sequences using random search or any standard planning algorithm. Once we have a set of successful action sequences we can form abstractions by finding least general generalizations over this set. The abstractions can be used as high-level actions. This process can be repeated with plans containing high-level actions to get abstractions over abstractions. We still need descriptions for these high-level actions in order to properly plan with them.

The idea for learning the description of an HLA is to view the description as an abstraction over the set of reachable states for the HLA. We can generate a set of reachable states by using the HLA repeatedly from different starting states and recording the resulting states. Once we have a set of reached states we can create a compact description by finding least general generalizations over this set. These descriptions along with their respective high-level actions can be used with the angelic hierarchical planning algorithms.

5 Timeline

- implement the warehouse domain and original angelic hierarchical planning algorithm, design so it will be easy to retrieve action sequences and start/end states for actions (1 month)
  - implement the warehouse world with primitive actions (7 days)
  - implement high-level actions and descriptions (9 days)
  - implement hierarchical planning algorithm (14 days)

- experiment with applying abstraction to action sequences and flesh out algorithm for learning HLAs (1 month)
  - try to recover a single HLA from traces generated by a true HLA (14 days)
  - try to recover multiple HLAs from traces generated by a sequence of HLAs (11 days)
  - try to learn HLAs from full traces of a problem where there is no "ground truth" i.e. learn from long primitive action sequences (5 days)
- experiment with applying abstraction to initial/reachable state sets and flesh out algorithms for learning semantics of HLAs (1 month)
  - learn description for a primitive action (15 days)
  - learn description for a HLA (15 days)
- put everything together and apply learning to the problem in the original warehouse world (1 month)
  - learn HLAs for the original warehouse problem (13 days)
  - compare learned HLAs to hand-designed HLAs (4 days)
  - learn HLAs on one instance then on another instance compare angelic planning with learned HLAs to standard planning algorithms (13 days)