Some Key Developments in AI Planning

Artificial Intelligente Nanodegree

Celso Araujo

December 16th, 2017

This is a short review on three papers on the Planning and Search sub-field of Artificial Intelligence:

- [1] Davis, Martin; Logemann, George; Loveland, Donald, "A Machine Program for Theorem Proving". Communications of the ACM. 5 (7): 394–397, (1962)[1]
- [2] Richard E. Fikes, Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", (Winter 1971)[2]
- [3] A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis", Artificial Intelligence, 90:281--300 (1997)[3].

The first paper, as suggested by its title, describes the implementation of an algorithm for theorem proving under propositional logic. A theorem is presented as a set of axioms together with a conclusion. Axioms and conclusion are sentences (called by the authors as “well-formed formulas”) which could either be an “atomic” sentence or a formula. An atomic sentence is just the statement of a property or its negation, and a formula is a sequence of atomic sentences put together by parenthesis and the operators ~, **&** and ∨ (“not”, “and”, “or”). If-then type sentences can be written by noting that **A** → **B** is the same as ~**A** ∨ **B**. The authors used 0-1 matrices to encode the sentences, and suggested a procedure based on repeatedly applying de Morgan laws on the axioms and new sentences then formed, in order to get terms canceled out, and the desired result to be reached. An example is provided, where the authors prove a simple theorem in algebraic group theory using this automated approach.

On the second paper, the authors describe a problem-solving program called STRIPS. This program takes ideas from theorem provers and aim to apply them to solve problems which can be encoded in first-order logic sentences. We have initial states (called “world model”) and allowed actions, and both can be viewed as forming the axioms, and the goal state is the conclusion. The authors suggest using tree search algorithms to solve the problems (or: to prove a theorem), by repeatedly applying allowed actions to given states, until either the goal state is reached or the problem is shown to be unsolvable. The authors describe as an example a problem where a robot has to push boxes from initial places to desired final places, turn a light switch on and go to another room.

The third paper introduces a planner called Graphplan. It acts on a STRIPS-like domain, with a problem being described by initial states, allowed actions and a goal state. The main idea here is to iteratively build a graph having two alternating layers. It begins with the initial state, where each individual atomic state is a node. This is the first layer, a states layer. Allowed actions are then applied to the state, forming an actions layer, and the resulting states set forms another states layer. This goes on until either the goal state is reached or the problem is shown to be unsolvable (a way to verify unsolvability is if the resulting layers start to repeat themselves). Graph search algorithms and heuristics can then be applied to determine a plan (ie, a set of actions connecting initial and goal states) – by iteratively building the graph – hopefully and preferably finding an optimal plan (ie, the shortest one). It is interesting to note that Graphplan suggests considering mutually exclusive relations (“mutex”) between nodes as a powerful way to help reducing the search subgraph, thus making computations faster. Finally, Graphplan takes only polynomial time to be built. The paper shows succesfull applications of Graphplan to the classical “rocket”, “flat tire” and “monkeys and bananas” problems.

These three papers outlined here show historical developments on the field of AI planning, that is, in determining how to find a feasible – and hopefully optimal – set of actions to take an initial state to a desired final state, for problems that can be modeled using boolean logic. The connection of planning to theorem proving is strong, and one could say that planning originates from theorem proving: [1] shows an automated theorem prover, [2] takes ideas from theorem proving to build a problem solver (a planner), and [3] shows an efficient way to consider – and implement – planning (and thus theorem proving) as a graph search problem.

1 Taken from http://www.divms.uiowa.edu/~tinelli/classes/295/Spring03/papers/Dav62.pdf
2 Taken from http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/strips.pdf
3 Taken from https://www.cs.cmu.edu/~avrim/Papers/graphplan.pdf