Learning of Agents with Limited Resources

Sławomir Nowaczyk
Department of Computer Science, Lund University
Box 118, 221 00 Lund, Sweden
Slawomir.Nowaczyk@cs.lth.se
http://www.cs.lth.se/home/Slawomir_Nowaczyk/

Abstract

In our research we investigate rational agent which consciously balances deliberation and acting, and uses learning to augment its reasoning. It creates several partial plans, uses past experience to choose the best one and, by executing it, gains new knowledge about the world. We analyse a possible application of Inductive Logic Programming to learn how to evaluate partial plans in a resource-constrained way. We also discuss how ILP framework can generalise partial plans.

Introduction

The idea of our project is to investigate rational agents, situated in an environment, that are able to consciously alternate between reasoning and acting, and complement their deductive abilities with knowledge extracted from experience.

Up to now our work has been focusing on agents that deal with planning in domains where complexity makes finding complete solutions intractable. Obviously, there are many such domains which are interesting from practical point of view and where it is not realistic to expect an agent to be able to create a complete plan for solving a problem at hand.

Thus, it can be interesting to let an agent create and reason about *partial plans*. By that we mean plans which are manageably short, but that still bring it somewhat closer to achieving the goal. Moreover, in order to make informed decisions about balancing deliberation and acting, the agent needs to be *time-aware*. In our work the main focus has been on plans which provide additional knowledge to the agent.

By creating and executing "information-providing" partial plan, agent can simplify subsequent reasoning — there is no need to analyse the vast number of possibilities which will be inconsistent with newly observed state of the world. Thus, it can proceed forward in a more effective way, by devoting its computational resources to more relevant issues.

Our goal is to create an agent able to function in an adversary environment which it can only partially observe and which it only partially understands. In order to succeed in this, the agent must be allowed to experiment for a number of episodes, learning from its mistakes and improving itself.

The testbed setting we envision is similar to the General Game Playing competition, where an agent is presented with rules of a previously unknown game and expected to start playing it effectively right away.

Architecture

The architecture of our agent consists of three main elements, as presented in Figure 1. The Deductor performs deductive reasoning about the world and agent's actions. Its main purpose is to generate a set of plans that could be executed in current situation and to predict, at least to some extent, the consequences of each of them.

The Actor chooses which one of those plans should ultimately be executed. It is worth mentioning that Actor also determines *when* to stop deliberation and initiate action.

Those two modules form the core of an agent. The idea is for the agent to move progressively closer to its goal, by sequentially executing such partial plans. At some point it should arrive at situation where Deductor can directly create a solution which provably reaches the goal.

However, success depends on whether Actor's choices are indeed moving the agent in the right direction. Since its knowledge can be incomplete and Deductor does not have enough computational resources to reach all logical conclusions, there is — in principle — no guarantee about that. In particular, it is possible that Actor makes a mistake, or a series of them, and the agent looses the game.

The third module in our architecture is supposed to remedy that. After an episode is over, regardless of whether the agent has won or lost, learning system attempts to inductively generalise the acquired experience.

We intend the learning to fill in gaps in the domain knowledge, to indicate generally interesting reasoning directions, to discover relevant subgoals and, finally, to allow Actor to choose the best partial plan more efficiently.

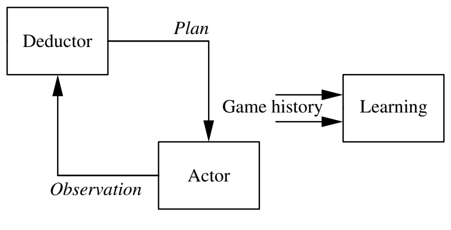

Figure 1: The architecture of the system.

Deductor

The Deductor uses First Order Logic, augmented with Situation Calculus mechanisms for describing action and change. In addition, predicate $Knows$ describes the agent's knowledge (Fagin *et al.* 1995), employing standard reification mechanism for using formulae as parameters.

A major concept in our formalism is a plan (a sequence of actions). And validity of a FOL formula depends not only on actual situation, but also on the plan an agent is currently considering. In particular, an agent can reason about formulae of the kind:

$$Knows[s, p, \alpha],$$

where α is a formula, s is a situation and p is a plan. Intuitively, it means: *agent knows that after executing, in situation s, plan p, formula α will be true*. Please observe that α can contain predicate $Knows$, thus allowing the agent to reason about knowledge-producing actions.

We employ Active Logic, a formalism describing the deduction as an ongoing process (Elgot-Drapkin *et al.* 1999), instead of characterising fixed-point consequence relation. It annotates every formula with a time-step label of when it was first derived, and bookkeeps every application of an inference rule by incrementing this label, e.g. $\frac{i:\ a, a \rightarrow b}{i+1:\ b}$

An important advantage of AL is its ability to handle inconsistency, which allows us not to distinguish between *knowledge* and *beliefs*. We assume that the agent's knowledge can be incorrect and even contradictory. Additionally, the *observation function* delivers axioms valid since a specific time-point and allows to model *external* events.

Reasoning about agent's own knowledge is crucial for us. Similar idea was introduced in (Petrick & Bacchus 2004), investigating how actions and observations modify agent's belief state and how such modifications can be propagated backwards and forwards through the history: as the agent gains knowledge, it can infer statements that *did* hold in past states of the world, even though it did not know it *then*.

Actor

Actor supervises the deduction process and interrupts it when, for example, it notices an interesting plan. It then evaluates partial plans and executes the best one of them, putting an agent in new situation, so Deductor can create new plans.

This is repeated as many times as needed, until an episode is finished. Losing the game clearly identifies bad choices and requires Actor to update its evaluation function. Winning the game yields important feedback for improving that function as well, but it also provides a possibility to (re)construct a more general plan. Even if new plan is not applicable to every situation, an Actor can still make use of it when evaluating other plans.

Learning

When analysing learning module, it is important to keep in mind that our agent has a dual aim, akin to the exploration and exploitation dilemma in reinforcement learning. On one hand, it wants to win the current game episode, but at the same time it needs to learn as much general knowledge as possible, in order to improve its future performance.

Currently we are mainly investigating the learning module from Actor's perspective — using ILP to evaluate quality of partial plans is, to the best of our knowledge, a novel idea.

One issue is that work on ILP has been dealing almost exclusively with the problem of *classification*, while our situation requires *evaluation*. There is no predefined set of classes into which plans should be assigned. What our agent needs is a way to choose the *best* one of them.

For now, however, we focus on distinguishing a special class of "bad" plans, namely ones that lead to losing the game. Clearly some plans — those that in agent's experience *did* so — are bad ones. But not every plan which does not cause the agent to lose is a *good* plan. Further, not every plan that leads to *winning* a game is a good one. An agent executing a dangerous plan might have just been lucky.

We define as positive examples plans which lead — or can be proven to *possibly* lead — to defeat. On the other hand, those plans which can be proven to *never* cause defeat are negative examples. There is a third class of plans, when neither of the above assertions can be proven. We are working on how to most effectively use such examples in learning.

However, this is only the beginning. After all, in many situations a more "proactive" approach than simple *not-losing* is required. One promising idea is to explore the epistemic quality: an agent should pursue those plans which provide the most important knowledge. Another way of expressing distinction between good and bad partial plans, one we feel can give very good results, is discovering relevant subgoals and landmarks, as (Hoffmann, Porteous, & Sebastia 2004).

Conclusions

We have introduced an agent architecture facilitating resource-aware deductive planning interwoven with plan execution and supported by inductive, life-long learning. The deduction mechanism used is based on Active Logic, in order to incorporate time-awareness into the deduction itself.

Currently, we are working on inductive generalisation of plans and on reusing a successful plan in different contexts. We are also looking into discovering subgoals and landmarks. In the future we intend to inductively discover general rules extending and correcting domain knowledge.

More detailed description of our research can be found at `http://www.cs.lth.se/home/Slawomir_Nowaczyk/phd-desc/`

References

Elgot-Drapkin, J.; Kraus, S.; Miller, M.; Nirkhe, M.; and Perlis, D. 1999. Active logics: A unified formal approach to episodic reasoning. Technical Report CS-TR-4072, University of Maryland.

Fagin, R.; Halpern, J. Y.; Vardi, M. Y.; and Moses, Y. 1995. *Reasoning about knowledge*. MIT Press.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Petrick, R. P. A., and Bacchus, F. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2–11.