Relevance Based Planning: Why Its a Core Process for AGI

Eric B. Baum

Baum Research Enterprises
41 Allison Road
Princeton NJ 08540
ebaum@fastmail.fm

Abstract

Relevance Based Planning (RBP) is a general method that plans in interaction with a domain simulation and domain specialized procedures. I argue that exploitation of the properties of causality and Euclidean topology which hold in many domains is a critical inductive bias necessary if an AGI (or any intelligent program) is to generalize to new problems and new domains, and is critical to human thought, and that RBP achieves its efficiency by exploiting these properties in a novel and powerful way, faithful to introspection in an example task. RBP is proposed to be implemented as a scaffold within an AGI or a CAD tool for producing intelligent programs, that is to say as a general library procedure that takes as inputs domain specialized procedures such as a domain simulation and procedures for recognizing and dealing with problems within the simulation. Supplied with such procedures as inputs, the RBP scaffold provides a framework that orchestrates plan formation and refinement in such a way that only causally relevant configurations are considered.

Introduction

The critical question in AGI is generalization: how one can generalize to solve new problems never before seen. Consider figure 1. You may never have seen this Rube Goldberg image before, but given a few minutes you can work out how and whether it works, and if there were a problem could suggest a fix. More generally, you can learn in a few minutes or years, to solve generic problems in whole new domains, for example new games or new employment. The learning is remarkably fast, given the complexity of the endeavor, and once one has learned, the solution often happens in real time. Each such new domain (or, arguably, new problem) requires rapidly constructing a novel, powerful program within your mind and executing it to solve new problems. I suggest that the only way this is possible is because you exploit in constructing and applying these mental programs certain underlying structure and properties of the domains (causality, Euclidean 3-D topology, etc); for example to analyze the Rube Goldberg device, that you construct and run a mental simulation of the system, and that doing this requires retrieving and rapidly putting together special purpose *procedures* that know how (or at least roughly how) to exploit causality and apply mental simulations, and procedures to analyze local structure, for example methods to simulate bird flight or ball rolling. The solving process follows causal chains, and may never bother to analyze the faucet because causal chains never reach it. An extensive literature indicates that human reasoning is model based (cf (Johnson-Laird 1983).

I further conjecture (a) that a program to perform such feats is naturally composed of scaffolds that take specialized procedures as arguments, for example a scaffold for handling the causal propagation and the interaction with a simulation domain, that takes as arguments procedures such as for recognizing and dealing with specific kinds of obstacles in specific domains. If one achieves a breakup like this, into general scaffold and specialized[1] procedures as inputs, then one can imagine gaining a concisely specified program that can generalize to rapidly construct programs dealing with new circumstances. And I conjecture (b) that the structure of such scaffolds can be informed by introspection.

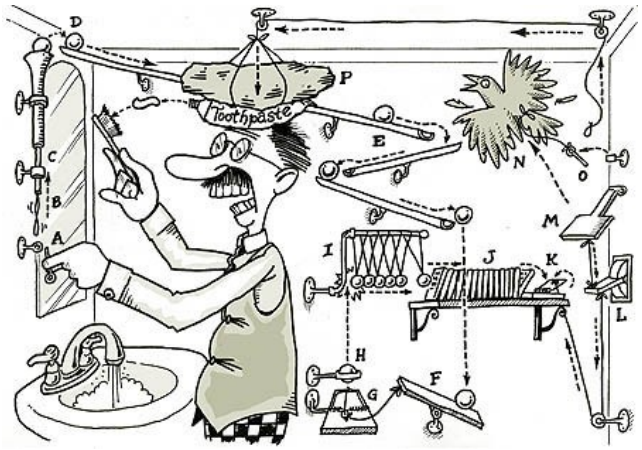

Figure 1: A Rube Goldberg Device

Relevance Based Planning(RBP) is a procedure that attempts to reproduce these aspects of thought. It is unlike any planning algorithm that I am aware of in the literature in its use of a simulation model and special purpose

[1] I refer to these as domain specialized to reflect the they might be adapted or inherit from more general procedures, which would further facilitate learning and generalization to new domains.

procedures for simulating or dealing with local structures, and in its fidelity to introspection at least in examples of Sokoban where I have experimented. It was designed by initially introspecting how I solve simple problems within Sokoban and then abstracting the procedure to a high level process that handles the causality, use of simulation domain, and backup when problems are encountered, by use of domain specific procedures recognizing and dealing with various kinds of domain specific objects.

Here is some high level pseudo-code for RBP:

(1) Find high level plans by search or dynamic programming over a simulation model, in which the search is over sequences of domain-appropriate operations, for example plans that allow counterfactual steps (steps that apply an action or stored procedure that would be applicable only if some obstacle in the simulation model may be corrected, where the obstacle is of a type that is known may be remediable). A high level plan then consists of a sequential-in-time series of subgoals (or obstacles to be overcome) that if solved should result in a solution.
(2) Refine one of these candidate plans in time order.
(3) As each obstacle in the plan is considered, invoke a (specialized) method that attempts to deal with the obstacle. The method typically performs some search on the simulation domain (and may invoke or even recursively call RBP).

Such calls to clearing methods typically pass information about higher level goals within the plan, and the called clearing method may then avoid searching a set of actions to remove the obstacle that would have as prerequisite previously achieving a higher level goal.
(4) If the search encounters other problems that would require earlier actions, (problematic configurations in the simulation domain that are perceived by agents for recognizing them) insert earlier in the plan an attempt to perform those actions first (typically by invoking a stored method for dealing with that kind of obstacle).
(5) When changes are made in the domain simulation, mark them, so that if they encumber later actions, you can back up and try to insert the later actions first to avoid the problem.
(6) Utilize a method of backing up to other high level plans when it is discovered that a high level plan can not be made to work, or of switching between high level plans as more information is uncovered about them.

RBP illustrates the power of using a domain simulation. It forms a high level plan over the simulation. Then it analyzes it in interaction with the simulation. As modules are executed to solve problems, the interaction with the simulation creates patterns that summon other modules/agents to solve them. This all happens in a causal way: things being done on the simulation cause other things to be done, and because the simulation realizes the underlying causal structure of the world, this causal structure is grounded. That is, it corresponds to reality. The RBP framework also focuses the search to consider only quantities causally relevant to fixing problems.

Many planning methods choose not to work in time order for various good reasons(Russell and Norvig 375-461). But by working on a candidate plan in time order, RBP is assured, at the inner most loops of the program, of only spending time considering positions that can be reached, and which are causally relevant to a high level plan. Introspection sometimes works out of time order on causally local regions of a problem, which are later interleaved in a causal (and time-sequential) fashion. As discussed in (Baum 2008a) this can be to an extent handled within RBP at a higher level.

(Baum 2008a) describes RBP in more detail, giving a step by step walk-through of a particular example of the application within the domain of Sokoban, and another example (at a higher level goal within Sokoban) is sketched. To formalize the pseudo-code to any given domain or problem within a domain, requires supplying various procedures that say what the obstacles are and how they are dealt with, what deadlock configurations look like, what counter-factuals are permitted, etc. In complex situations it is expected that some of these will be too hard to be hand coded, and will instead be produced by learning from examples using module constructors such as Evolutionary Economic Systems (Baum, 2008b).

If an RBP scaffold is provided, it can be made accessible to module constructors within an AGI or CAD tool, so that new programs can be automatically constructed, evolved, or learned that invoke the RBP. An example where an RBP planner was used in this way was given in (Schaul 2005).

A larger meta-goal of the project described in (Baum 2008b) is to construct Occam programs, programs that are coded in extremely concise fashion so that they generalize to new problems as such problems arise (or so that the programs can be rapidly modified, say by a search through meaningful modifications, to solve new problems). Search programs can be incredibly concisely coded, so that coding as an interaction of a number of search programs can be a mode of achieving great conciseness. RBP's construction of a program by composing a series of feasible-sized goal oriented searches mirrors the basic structure of biological development (cf (Baum 2008b)), and is similarly concisely coded and robust.

References

Baum, Eric B. 2008a. *Relevance Based Planning: A Worked Example.* http://www.whatisthought.com/planning.pdf .

Baum, Eric B. 2008b. *Project to Build Programs That Understand.* Proceedings of AGI09 (this volume) http://www.whatisthought.com/agipaper091.pdf .

Johnson-Laird, P. 1983. *Mental Models.* Harvard University Press, Cambridge Ma.

Russell, S. and P. Norvig 2003. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Saddle-River NJ.

Schaul, T. 2005. *Evolution of a compact Sokoban solver.* Master Thesis, École Polytechnique Fédérale de Lausanne. posted on http://whatisthought.com/eric.html