MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 303 September, 1987

PLANNING IS JUST A WAY OF AVOIDING FIGURING OUT WHAT TO DO NEXT

Rodney A. Brooks

Abstract. The idea of planning and plan execution is just an intuition based decomposition. There is no reason it *has to be* that way. Most likely in the long term, real empirical evidence from systems we **know** to be built that way (from designing them like that) will determine whether its a very good idea or not. Any particular planner is simply an abstraction barrier. Below that level we get a choice of whether to slot in another planner or to place a program which *does the right thing.* Why stop there? Maybe we can go up the hierarchy and eliminate the planners there too. To do this we must move from a state based way of reasoning to a process based way of acting.

[**McDermott 1987**] recently asked the following questions in regard to the need to do resarch in interleaving planning and run-time monitoring of senors:

Are sensors good enough yet for us to be thinking about how to reason about their inputs? Is planning research just a typical AI moonshine enterprise, thinking about hypothetical scenarios that don't resemble what will actually be possible once high-quality sensors are available? We can distinguish three positions on this question:

Theism: *Execution monitoring is important enough, and the issues clear enough, for us to be thinking about them right now.*

Atheism: *The whole idea of plan execution and the run-time maintenance of something called a "plan" is misguided. Controlling the behavior of a robot is a matter of putting sensors and effectors together using a program.*

Agnosticism: *We won't be able to settle the issue until much better sensor technology is available.*

This is my reply to McDermott.

I am an atheist in McDermott's sense:

An atheist usually has very little hope of convincing a theist of his folly. The theist afterall has his own self consistent set of beliefs. Likewise I expect to make little progress arguing with planning theists. I expect they will make little progress arguing with me.

But religous theists and atheists have an extra problem to deal with. The only possible resolution of the debate involves one of the participants dying, and even then at most one of them learns the true facts. In the matter of plan execution, however, we have an empirical test available! We can try to build real robots that operate in real environments and see which ones work better and appear more intelligent; those with traditional AI planning and execution monitoring systems or those with reflexive or subsumption architectures. One of us will have to eat our words!

But will this happen soon; aren't our computers too small yet? Heck no. Part of my thesis is that it actually takes very little computational power; we've just been organizing it all wrong up until now.

1. Wrong decomposition.

I'll start with some seemingly irrelevant fables. However, I do believe they are precisely relevant and applicable to the debate about planning and execution.

Fable 1: How does a FORTRAN computer work?

Once upon a time in a land far away a young boy was given a book on FORTRAN programming. The boy had never seen a real computer, nor had any idea how one worked or what it really did, but he had seen pictures of "giant brains" in '50s juvenile science books along with a discussion of binary arithmetic, and simple relay-based switching circuits.

He started reading the FORTRAN book and was immediately convinced of the power of computers. He wanted to program one. And he started thinking about how one might be built.

It was pretty clear how to break the problem down into functional units. There must be a method of encoding the characters on the FORTRAN coding sheets into binary numbers which then control the switching paths in the rest of the computer. There also must be a big set of memory registers (he thought of them as little cubby holes that you could put numbers in). One sort for REAL numbers (about twenty twenty-sixths of them) and one sort for INTEGER numbers (the other six twenty-sixths). Actually there must be a lot of these registers; one named A, one named B, one named XYZZY, one name B365QL, etc. In fact there must be $26 \times 36 \times 36 \times 36 \times 36 \times 36$ of them. He never bothered to multiply it out, but, gee, there's an awful lot of them. Maybe thats why computers cost over a million dollars (and remember this is when a million dollars was real money!). Oh, and then there must be a set of circuits to do arithmetic (pretty easy to see how to do that) and a way of getting numbers back and forth to the cubbyholes.

So that was how to build a computer. Then the boy read the chapter on arrays and did a few multiplications with pencil and paper. Oh, oh!

Fable 2: OK, How does a FORTRAN computer really work?

In another universe some neuroscientists were given a FORTRAN computer. They wanted to figure out how it worked. They knew that in running a porgram there were three different processes: compilation, linking and loading. They decided to first isolate in the machine, subareas where each of these things were done. Then they'd be able to study each of them in isolation and in more detail. So they got some oscilloscopes and started probing the wires and solder points within the machine as it ran programs, trying to correlate the signal trains with what the operator console reported was happening.

They tried for many years but activity throughout most of the circuits from a global perspective seemed uniformly like white noise. Locally they could establish correlations, but not correlate that with what the console reported. The only susbstantial clue was that a rotating mechanical device made a different pattern of noises when the different computer behaviors were dominant. One popular hypothesis was that all the real computing was done mechanically in thus unit somehow and that all the electrical circuits where just a routing network to get stuff to and from the I/O devices. But there were other hypotheses too. Maybe the computer was using holograms.

The point.

The point of these fables is that without having designed a device yourself, or thought through a design completely, you may very well make completely the wrong functional decomposition by simply observing its behavior. The same is true of observing human behavior. Ethologists have discovered this in observing insects and lower animals. Early and folk or intuitive explanations of what the creature is doing have had to undergo radical change when more careful observation and experiment with the total system (creature and environment) have been carried out.

The idea of planning and plan execution is just an intuition based decomposition. It may well be the wrong decomposition. There may be no reason it *has to be* that way. All we have to go on is our intuition of how we work—historically that intuition has been wrong. Most likely in the long term, real empirical evidence from systems we **know** to be built with planners and plan execution modules (from designing them like that) will determine whether its a very good idea or not.

2. What's wrong with models and plans?

Plans provide a useful level of abstraction for a designer or observer of a system but provide nothing to a robot operationally.

If you allow the notion of explicit plans for a robot you run into a problem of which level of abstraction should the plan be described at. Whatever you decide, upon examination it will turn out to be a bogus and arbitrary decision.

How much detail?

Consider a mobile robot which must cross from one part of a room to another. A traditional AI planner would ignore the geometry of the room and simply have a list of known named places and would issue some plan step like *MOVE from A to B*. Then it is up to some assumed runtime system to execute this step, and perhaps re-invoke the planner if it fails for some reason. In simulation systems the runtime system typically achieves such plan steps atomically. But when many AI-roboticists came to implement the runtime systems on physical robots they found they needed to use a planner also (e.g. [**Moravec 1983**]). This planner takes into account a floor plan modelled from sensor data and plans a collision free path for the robot (and usually it is only intended for static environments, not those with people moving about in them). Some such robots then send the individual path segments off to subroutines which execute them. Others rely on yet another level of explicit planning to decide on how to accelerate the motors, perhaps based on a model of the mechanical dynamics of the vehicle. All such planners that I know of then pass a series of commands off to some lower level program which *does the right thing*; i.e., it takes the action specification and directly translates it into signals to drive motors. One could imagine however, yet another level of planning, where an explicit model of the drive circuits was used to symbolically plan how to vary the motor currents and voltages!

We thus see that any particular planner is simply an abstraction barrier. Below that level we get a choice of whether to slot in another planner or to place a program which *does the right thing*. What could this mean? Let's look at some examples from robotics.

Previous examples from Robotics.

Below are two pertinent examples from robotics. In both cases early attempts to control a robot by telling it how to set its joints in space have been replaced by telling it the parameters to use in tight feedback loops with its environment. The controlling program no longer tells it, nor knows, nor cares, where to set its joints. Rather, in each case, the robot acts differentially to the environment, trying to maintain set points in a higher order space. As a result of interaction with the environment the overall goal is achieved without the robot controller ever knowing the complete details of how it was done.

In the early days of research robotics, and even today in most of industry it was assumed that the right abstraction level for talking to a manipulator robot was to tell it to go someplace. The driver program, or planner perhaps, sends down a series of desired locations and orientations for the end effector (some systems operate in joint coordinates, while others operate in cartesian coordinates). This is known as *position control*.

Experience over many years has shown some problems with this approach. First, in order to interact well with the world, the world model must have extremely high precision; making sensing difficult and expensive. Second, in order to carry out tasks with low error tolerances, it is necessary to have precise position control over the robot; clearly making it more expensive. But since position accuracy is critically dependent on the available dynamic model of the robot and since grasping a payload alters the manipulator dynamics we have seen a trend towards more and more massive robots carrying smaller and smaller payloads. Manipulator to payload mass ratios of 100 to 1 are almost the rule and ratios of 1000 to 1 are not unheard of. Compare this to the human arm.

Recently researchers have realized that position control was not the best way to approach the problem. Now research abounds on *force control* and ways to use it (automatically) to achieve delicate and precise goals. The major idea in force control is that rather than tell the robot to achieve some *position* it is instead told to achieve some *force* in its interaction with the environment. As long as the desired force is appropriate for the circumstances, the physical interaction of the robot and the world guide it to achieving some desired goal. For instance holding a peg tilted at an angle and applying a force outside the friction (or sticking) cone, the peg slides across a surface until the lowest corner drops into a tight fitting hole, whereafter the peg slides down the hole. Much tighter tolerance assemblies can be achieved in this way than with the use of pure position control. Indeed many of the force control strategies used tend to be similar to those used by humans.

Notice that at no time does any program have to know or care precisely where the robot will be. Rather a "planner" must arrange things so that the constraints from the physical world are exactly right so that the robot, operating so as to do the specified right thing, can't but help achieve the higher level goal. The trick is to find a process for the robot which is stable in achieving the goal over a wide range of initial conditions and applied forces. Then the world need not be modelled so precisely and the manipulator dynamics need not be known so precisely.

In a second example of this trend, Raibert [**Raibert et al 1984**] has elegantly demonstrated that the intuitive decomposition of how to walk or run is maybe not the best. Most previous work in walking machines had concentrated on maintaining static stability and carefully planning where and when to move each leg and foot. Raibert instead decomposed the running problem for a one (and later two and four) legged robot into one of separately maintaining hopping height, forward velocity and body attitude. There is certainly no notion of planning how the running robot will move its joints, where exactly the foot will be placed in an absolute coordinate system, or where the body will be in six dimensional configuration space at any give time. These questions do not even make sense within the decomposition Raibert has developed.

Both these are example of redefining the *right thing* is in a way that radically redefines the "planning" problem.

What does this all mean?

[**Brooks 1986**] has shown that there is another way to implement *MOVE from A to B*.* A simple difference engine forces the robot to move towards *B* while other parallel activities take care of avoiding obstacles (even dynamic ones). Essentially the idea is to set up appropriate, well conditioned, tight feedback loops between sensing and action, with the external world as the medium for the loop.

*Although his work suggests that this is not an appropriate subgoal to be considered for a higher level plan.

So it looks like we can get rid of all the planners that normally exist below a traditional AI planner. Why stop there? Maybe we can go up the hierarchy and eliminate the planners there too. But how can we do this?

We need to move away from state as the primary abstraction for thinking about the world. Rather we should think about processes which implement the *right thing*. We arrange for certain processes to be pre-disposed to be active and then given the right physical circumstances the goal will be achieved. The behaviors are gated on sensory inputs and so are only active under circumstances where they might be appropriate. Of course, one needs to have a number of pre-disposed behaviors to provide robustness when the primary behavior fails due to being gated out.

As we keep finding out what sort of processes implement the *right thing*, we continually redefine what the planner is expected to do. Eventually we won't need one.

3. Simulation.

[McDermott 1987] also asks whether it is sufficient to pursue these questions using simulated systems. The answer is a clear no.
I support the use of simulation as an adjunct to real world experiments. It can cut development time and point up problems before expensive hardware is built. However it requires a constant feedback from real experiments to ensure that it is not being abused.

The basic problem is that simulation is a very dangerous weapon indeed. It is full of temptations to be mis-used. At any level there is a temptation to over idealize just what the sensors can deliver. Worse, the user may make a genuine mistake, not realizing just how much noise exists in the real world. Even worse than that however, is the temptation to simulate the 'perception' system. Now we jump right into making up a decomposition and stating requirements on what the perception system will deliver. Typically it is supposed to deliver the identity and location of objects. There exists no computer perception system today that can do such a thing except under very controlled circumstances (e.g., a small library of machined parts can be localized and recognized in a small field of view from a fixed camera). I don't believe a general such system is even possible; for instance I don't believe humans have such a perception system. The idea that it is possible is based on the wrong-headed decomposition that gives us planning systems.

Don't use simulation as your primary testbed. In the long run you will be wasting your time and your sponsor's money.

Bibliography.

[Brooks 1986] "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April, 14–23.

[McDermott 1987] Drew McDermott, *position paper for DARPA Santa Cruz panel on interleaving planning and execution*, October.

[Moravec 1983] "The Stanford Cart and the CMU Rover", Hans P. Moravec, *Proceedings of the IEEE*, 71, July, 872–884.

[Raibert et al 1984] "3-D Balance Using 2-D Algorithms?", Marc H. Raibert, H. Benjamin Brown, Jr., and Seshashayee S. Murthy, *Robotics Research 1, Brady and Paul eds, MIT Press*, 215–224.