LECTURE 1: INTRODUCTION

OK, let's begin: WHAT IS AI?

Well, this is a bit hard to say. It started out (some say in 1956) in part at least, as an attempt to come up with a COMPUTATIONAL UNDERSTANDING OF GENERAL-PURPOSE HUMAN-LEVEL COGNITIVE ABILITIES, esp regarding language, reasoning, understanding, decision-making, etc. Some version of this idea had been around for a long time, perhaps even millennia. We'll look at a little more history later on. But with the advent of computers it soon became a recognized group endeavor, and in 1956 ten scientists spent a summer together at the now-famous Dartmouth Conference to discuss just that. Among them were John McCarthy (who coined the phrase Artificial Intelligence), Marvin Minsky, Alan Newell, and Herbert Simon -- they quickly became the leading lights of this new field and shaped it for many years.

It was soon discovered that this is a lot harder than it seemed, and eventually people began to focus on particular narrow topics where they could make progress, such as automated planning, computer vision, expert systems, machine learning, automated theorem-proving, and so on -- and for many decades the original vision was almost forgotten. But more recently it has resurfaced, largely in the form of investigation of so-called intelligent agents.

We will study AI largely by using a few illustrative agent-examples, starting in particular with this one:

ROBBIE, THE HELPFUL ROBOT: we are studying for an exam which starts in less than an hour, and it occurs to us that it might help to have a particular book B to consult. But we don't have time to go get book B ourselves -- we are memorizing some crucial terms. So we ask our robot friend Robbie to find the book for us, suggesting that it might be in room X.

What abilities does Robbie need, to accomplish this task? That of being able to get book B from room X is far too specialized -- it's not sensible to build a robot for just that one purpose. So Robbie needs general-purpose abilities, such as locomotion from one location to another, planning, vision, reaching, error-correction, reading, learning, etc. In fact, this simple-seeming problem is most likely AI-complete: a reliable general solution probably involves solving just about all of the open research issues in AI!

That is, the present state of the art is well below that needed to build such a robot (and associated software). But we can study many of the particular abilities Robbie would need, and assess how far along we are with regard to them. This will motivate much of what we study during the semester.

But we are getting ahead of ourselves. What is an AI problem more generally? Well, that's the topic for Thursday's lecture. But we can say that -- following the cue of our textbook -- it takes an agent to solve an AI problem, and an agent is...

... an implementation of a function from a percept-stream to an action-stream (acting within a given environment **E**):

Figure 1 shows this interaction.

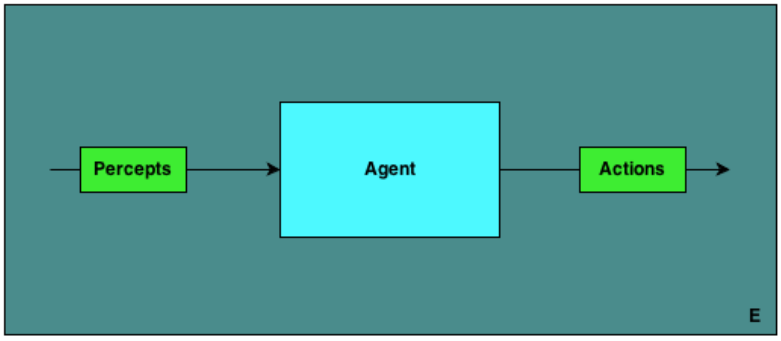

Figure 1

The textbook portrays this, as shown in Figure 2 below (Fig 2.1 in the textbook), a little differently, with E being external to the agent; an interesting but perhaps not fundamental difference.

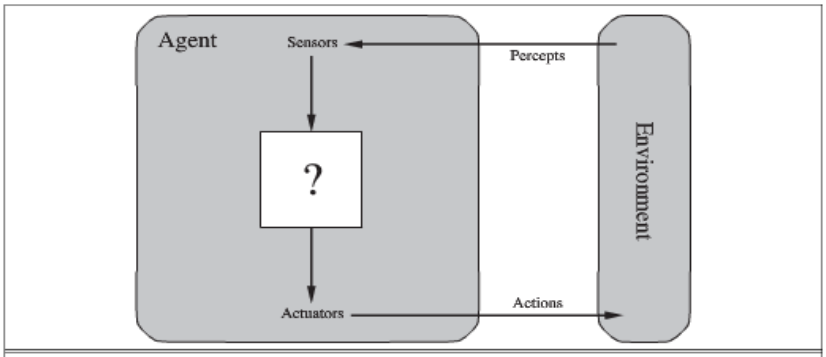

Figure 2

In general, given a particular (sequence of) percept(s), an agent may have a variety of possible actions; thus an agent amounts to a decision-process that selects (a sequence of) actions to perform, given input data (the percepts). This is incredibly general, and yet already leads to an important insight: an agent in a given environment has an associated graph (sometimes a tree) that can represent it's possible action-choice sequences over time.

An example of such a tree can be seen in Figure 3.

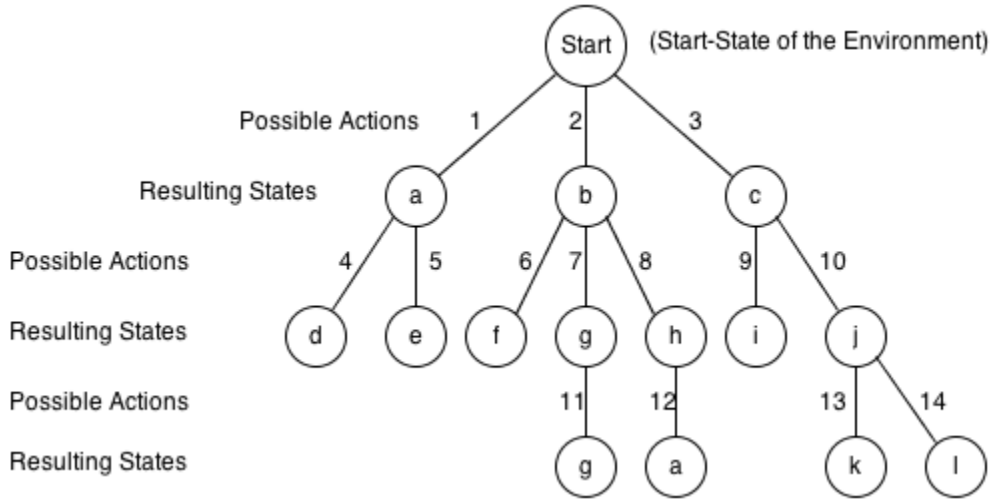

Figure 3

There are states and transitions in Figure 3. Transitions (or possible actions) are the lines (with numbers) going from state to state. The states are the circles with letters inside of them. From the start-state, action **1** leads to state **a** whereas action **3** leads to state **c**.

Note that g appears twice in Figure 3, as does state a. As it is drawn above, the graph is a tree (the so-called **search-tree**); but if we were to indicate action 12 instead as an upward arc from h to the a-state one level higher on the left, this would then no longer be a tree. The tree from above represented as a graph can be seen in Figure 4; this is called the **state-space graph.**

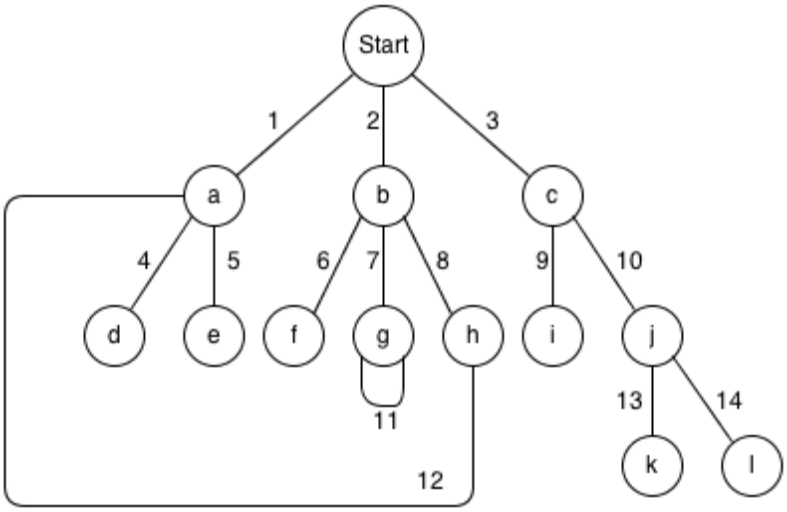

Figure 4

Each method is useful. Basically, in the search-tree version, paths from the root indicate possible sequences of actions where each action produces a new "node" that not only represents the state of the world that would result from those actions, but also the fact that those actions were done (the history); whereas the state-space version simply shows the possible states of the world and how one can get from one to another. Either can be reproduced from the other.

Now, given a goal-state to try to achieve (let's call it g), then what the agent (or function, or program) has to do is FIND A PATH from START to g. We assume (for now at least) that the agent can perceive the current state of the world at any moment, and also can assess the resulting new state of applying any available action to the current state.

In effect, the agent needs to SEARCH among possible paths, for one that reaches a goal-state. So, given a goal and possible actions per percept, what we have is a SEARCH-PROBLEM.

Moreover, we would like the agent to come up with a GOOD solution, one that gives the best performance (maximizes effectiveness) according to some given measure(s); such an agent is said to be RATIONAL. For instance, we might want the agent above to find the instance of g that is easiest/fastest/cheapest to get to.

So, in a nutshell, AI in general can be seen as a problem of searching a tree (or graph) for a path leading to a given goal. And for some purposes that is a very good way to look at it. Consider this problem: someone has stolen your pot of gold, and (we have reason to think) has buried it in the desert. How will we find it? If we have no other information to go on, we just start digging here and there, but hopefully not at total random. We'd like it to be methodical, so that (i) we don't gog in the same place twice, and (ii) we don't skip places forever. Let's say we can go North, South, East or West after each dig. We might try to go in a sort of spiral, widening out from a starting point: go North, then West, then South, then South again, then East, then East again, then North, North, North, West, West, West, etc.

How would we represent that as a search-tree? As a state-space graph?

Since there is no data to guide the search, nothing indicating higher or lower likelihood of the gold being at one place or another, then we have little choice but to search in a systematic but otherwise uninformed way. And this kind of search is called *uninformed* or *blind* search, But that is not to put it down – sometimes it is all we have, and it can be very effective. We will see various versions of this next time.

For now we turn to another basic concept.

THE IDEA OF AN "AI PROBLEM"

An AI Problem is, simply, a task that we might want an intelligent agent to solve -- eg our helpful Robbie. It involves a goal, possible actions and their (likely) results, as well as data/percepts, costs/benefits, and so on. However, in AI, these tend to be highly underspecified, so that there is no direct simple connection between input and output, unlike a typical program specification.

Examples

find a book
find a way to **get** (take hold of) a book once it is found
find a way to move puzzle pieces to achieve a particular pattern
find a pattern with certain properties
interpret visual data
answer a question
make something happen
prevent something from happening
be helpful
and so on...

These vary tremendously, and we won't consider them all right now. But the first four are good ones to look at more closely, since they reveal again the general problem of AI Search. Here are four famous examples

Ex. 1: Sussman's Anomaly (page 371)

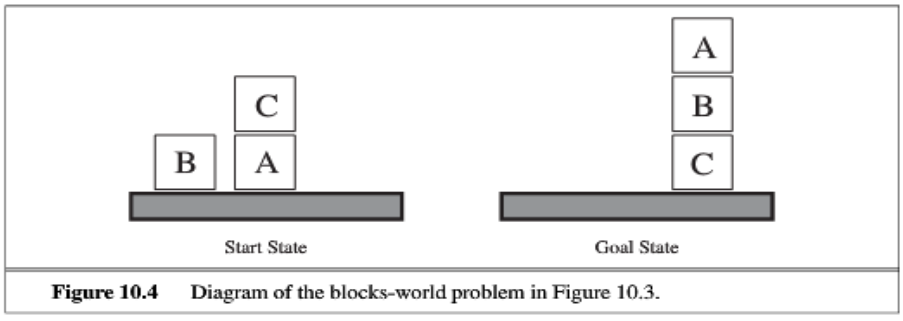

Figure 10.4 Diagram of the blocks-world problem in Figure 10.3.

Figure 1

Here the problem is to move blocks one at a time, to go from the start-state on the left to the goal-state on the right of Figure 1. It is trivial for a human to solve; but what knowledge and/or reasoning are we using to do it? If you try to spell it out in a general program (not designed for this one particular problem), it gets quite tricky. But it can be done as a search problem, where one specifies all possible actions in each state, and looks through the **state-space graph.**

Actions: The state-space graph for a problem consists of all the possible states the given environment can be in, and a directed edge connecting each state **s** to each state that a single applicable action can change **s** to. For Sussman's Anomaly, the actions are: move any block that has no block above it onto any other block that has no block above it, or onto the table. (In fact these are the typically actions in any so-called classical Blocks-World problem.)

States: Note that a *state* here is *not* the same as a drawing of block locations; state consists only in information (assertions) about whether a block is on the table or on a particular other block. This is a huge simplification over the complex information humans have available when reasoning about blocks, which is surely in large part why we can solve simple blocks-world problem effortlessly whereas crafting a program to do so is quite a chore in general. An example of a state is "B is on the table, A is on the table, C is on A."

For instance, how do we know that there is no block on C in the Sussman start-state? We can see there is none! But in the blocks-world, for a program's sake, we usually make the **closed-world assumption**: that if something is not explicitly asserted in the knowledge base, then it is not true. Since no block as asserted to be on C, then none is. What else do we humans so easily understand here? No one yet has anything like a complete understanding of this, but it includes things like the contextual possibilities for motion, falling, support – it's a whole dynamic reality of affordances that we see at a glance, rather than a static file of uninterpreted assertions. So there is lots of exciting work to be done, to get our programs on a par with humans here. On the other hand, when an example is very complicated (e.g., more blocks than we can take in visually), existing programs may do better than us in some respects.

– Food For Thought –

This speaks to the (varying) goals of AI. If we simply want programs that get a job done quickly and accurately—what could be called *engineering or technological AI* – then sometimes ignoring how humans do it may work just fine – maybe even better! But if part of our goal is what Hector Levesque (see his recent paper at http://www.cs.toronto.edu/~hector/Papers/ijcai-13-paper.pdf) calls "scientific AI" (or what I like to term *AI as a cognitive science*) then what we are after is an understanding of "intelligent behavior in computational terms". And intelligent behavior is not just getting results; it has to be done intelligently: taking lots of information/knowledge into account, reasoning with it, being aware of pitfalls, etc. Indeed, we don't have a specification in advance as to what the pieces are that make up intelligence; and this is the usual thing in science: we have an inkling of some phenomenon (living systems, chemical reactions, falling objects, forces) and grope our way along little by little.

An important variation on the state-space graph is the so-called **search-tree**. This is also a graph, with the start-state as root, but in which each application of an action to a state produces a new node at a lower level, even if the new state is actually one that has already been seen. Thus the nodes in a search-tree represent not only a state of the

environment but also the history of how it arose from a particular action applied to a particular earlier node (its parent). Note that in general, search trees tend to be infinite, with states being repeated over and over and lower and lower levels. Also note that, unlike the state-space itself, the search tree in general is not given in advance but has to be built (generated node by node) as it is examined.

Now this may sound pretty dumb: surely one does not need to blindly "search" through an infinite data structure until one hits by accident on the goal (A on B on C) or perhaps never finds it at all! And that is true. So-called blind (or uninformed) search – which we will examine in the next lecture – is often not a good way to search (although sometimes it is the only method available, when information that might help guide smarter search is just not available). In the Sussman case, there are better ways; but the best ways I believe remain to be discovered, involving a great deal of specialized world knowledge about blocks and support and motion and so on.

Ex. 2: Monkey and Bananas (pg 396 bottom; also in Homework 1)

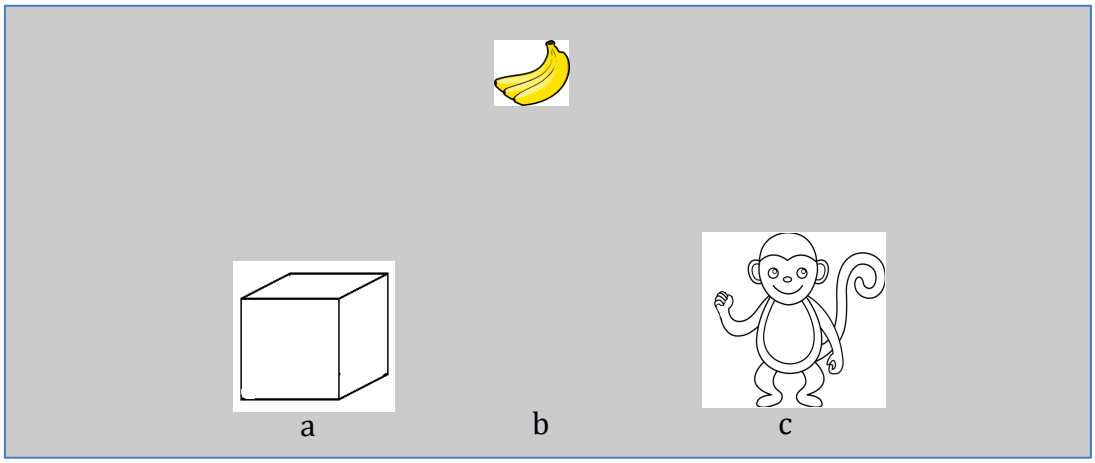

Figure 2

This famous problem was first published in a paper subtitled "The Hungry Monkey" in 1977 (see pdf link in this page:
http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog0102_2/abstract).

Figure 2 shows a bunch of bananas suspended from the ceiling above position **b** out of reach of a monkey (or robot) at **c**; and there is a box at position **a** which can be pushed and also climbed. Again, to a human the solution is obvious; but what exactly is our reasoning process here, and how can we capture enough of it in a program? If we specify allowed actions (push, climb, grasp, etc) we can again form a search space and begin to look through the associated search tree.

In this example, there is a more clear notion of agent: the monkey! And it clearly needs to formulate a plan to get the bananas. Solving this problem in a plausible and general way will take us at least a month of work! But doing so will reveal lots of powerful machinery.

Ex. 3: The 8-puzzle (pg 71)

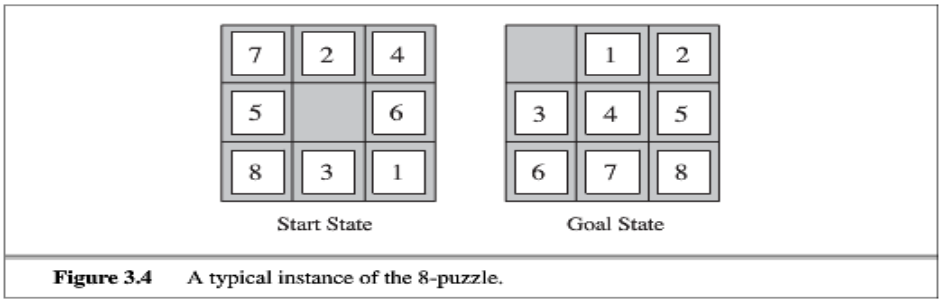

Figure 3.4 A typical instance of the 8-puzzle.

Figure 3

Figure 3 shows the 8-puzzle problem formulation with the start and goal states. There are many start and goal states for the 8-puzzle problem. Figure 3 shows only one example of a start-state and goal-state combination. This is a reduced version of the familiar 15-puzzle that people often encounter as kids: a 4x4 square (usually plastic or wood) is filled with 15 slideable tiles arranged in 15 of the 16 positions. The task is to rearrange them by sliding one at a time, into some specified goal pattern. Again one can draw the state-space and begin examining the associated search tree. Depending on the start and goal states, this can be short and easy, or long and tedious. It is a particularly good example to illustrate some of the basic AI search strategies, and we will do that in the next lecture.

Ex. 4: 8 queens (pg 72)

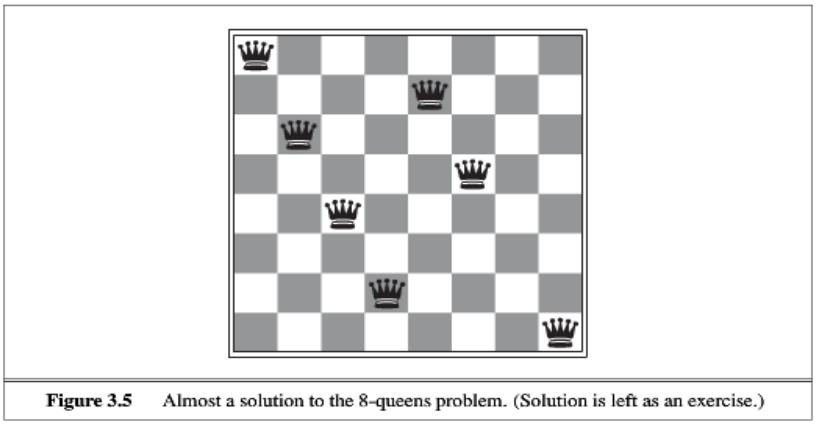

Figure 3.5 Almost a solution to the 8-queens problem. (Solution is left as an exercise.)

Figure 4

Figure 4 shows a chess-board with 8 queens positioned among the available 64 squares such that each is safe from attack by the others. (One of your homework problems considers the reduced case of 4 queens on a 4x4 board. More generally, one can study the n-queens problem, on an nxn board.)

Why is the 8-queens problem of a very different sort from all the others we have discussed so far? There are two reasons: (i) we did not really specify the goal state(s)! – all we did was say that a goal state has to satisfy a certain condition ("constraint"), so we don't really know what a goal might look like initially. And (ii) the whole problem is to *find out* just what a goal looks like – to come up with a goal state – and once we have done that, then *getting there* by some path is not really of interest (it is trivial to put the queens in place once we know where they go). Compare to the 8- or 15-puzzle, where we know where the tiles should go, but it is tough to figure out how to get them there by pure sliding actions.

A problem like n-queens – where the goals are not known in advance and the problem is to discover such states rather then how to get to them – are called **constraint satisfaction problems**, or CSPs. Of course, the n-queens problem can be reformulated so that only very specific actions are allowed (as in your homework problem) and then one can use traditional search methods to find a goal (by checking each new state in the growing search-tree to see if it satisfies the constraints).

What similarities do you see between these famous problems and the ones given just above them? What differences?

The four famous problems that we looked at are examples of so-called **toy problems**, vastly simplified, narrowed, unrealistic (except considered as games). What many AI researchers really want to be able to do is to solve problems such as these:

The (overall) NLP (natural-language processing) problem: given English (say) inputs, produce realistic and relevant English outputs.

The (overall) vision problem: given image data (a 2-dimensional array of pixel data), produce accurate English output describing the associated scene.

The (overall) learning problem: given time-series data, come up with patterns and concepts that accurately describe those patterns.

The (overall) planning problem: given data about a task and a situation, come up with a viable plan to perform the task.

The (overall) situational problem: given a situation, figure out – and do – the right (or at least a very good) thing, without causing any disasters in the process.

While huge amounts of progress have been made in all these areas within narrow constraints, solutions to the general problems – at a level comparable to human behavior – still seem well off in the future.

The next few lectures will focus on some important general search techniques. But it is worth pausing to ask: (i) is *search* really something an intelligent agent needs to be good at? And, (ii) while we can write search algorithms, and that is very useful, where is an agent involved in this?

Regarding (i) what we can say is that an intelligent agent very often (one perhaps could even say, almost all the time) is faced with choices as to what to do. And there are many options not only for what to do in the very next brief moment but also for what sequence of actions to be doing over time. That is, an agent needs to make plans in order to work intelligently toward achieving goals. (We'll return to question (ii) in a little bit.)

A plan is simply a sequence of actions. And among the many possible plans that there could be, the agent must have (at least) one plan to follow, in order for its actions to be coherent and useful. So, an agent needs to be able to select a plan out of many possible ones. That, more or less, is what AI search is all about. And as we have seen in the above examples, the way a particular "plan-search" plays out can vary tremendously from one situation (one problem) to another. Thus we have answered one question above: yes, an agent needs to be able to search for (find, come up with) a ("good") plan among possible plans. (Here "good" means *at least* that it will lead to a goal state. And that brings up a host of other issues, such as this: what could possibly *guarantee* that certain actions will result in what one expects? Ans: very often nothing! More on this later.)

– Food For Thought –

How about creating a plan, vs, merely finding one? Is this really a difference? In most cases, there is a given (or understood) finite set of possible actions, and then there is a given state-space graph (but it could be huge, and the search tree infinite). So it can take a lot of cleverness – or at least some sort of reasoning – to find anything at all in it. Perhaps that is what creativity is: being good at finding useful things out a bewildering array of possibilities. But the array is *there*, and the good possibilities are also there; we don't make them up out of nothing so much as we encounter them either by accident or clever searching. A truly new kind of action that we invent is very rare; more often we put available basic actions together in new ways, which is to say that we find an action sequence in the state space that we had not been aware of before.

What about the second question: given a search algorithm, where is the agent? And here we confess: there does not need to be an agent. It is entirely possible to have an excellent search algorithm, and run it usefully, with no agent involved (except the human who decides to run the algorithm). But just as a human may use a search algorithm to find a good plan (or path) in a complicated situation, so might a robot. Thus search algorithms are tools that an intelligent agent can use when the need arises. In fact, many AI

algorithms that are not in themselves search algorithms in fact employ search algorithms "behind the scenes" to make them run well. In the examples above, perhaps only the Monkey & Bananas makes crucial use of the agent as part of the planning/searching process. There, the agent needs to reason about not only other objects in the world, but also about its own role in those actions. (In fact, in the 8-queens problem there is no need even for a plan; all we seek is a detailed description of a goal state – unless of course one supposes that a plan is needed for how to go about an effective search for such a state! But as it turns out, there are ready-made/off-the-shelf search strategies that can be used.)

For now, let's try to outline some of the (many) abilities Robbie might need to do any or all of the above...

Vision (to see where it is going, and to see the room numbers, and to see the book, see its gripper holding the book, etc)
Language (to understand the task, to read numbers and titles, and to report back or to ask for help)
Reasoning (to be able to combine factual information to draw useful conclusions)
Planning (to be able to decide on a course of action)
Locomotion (to get from place to place)
Decision-making (not only to decide on a plan, but to decide on refinements or changes or whether a plan is failing and needs to be replaced, etc)
Learning (so it can retain new information for future use)
…and lots more.