Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Working Paper 267 February 1985

The Structures of Everyday Life

Philip E. Agre

Abstract: This note descends from a talk I gave at the AI Lab's Revolving Seminar series in November 1984. I offer it as an informal introduction to some work I've been doing over the last year on common sense reasoning. Four themes wander in and out.

1) Computation provides an observation vocabulary for introspection. With a little work, you can learn to exhume your models of everyday activities. This method can provide *empirical* grounding for computational theories of the central systems of mind.

2) The central systems of mind arise in each of us as a rational response to the impediments to living posed by the laws of computation. One of these laws is that all search problems (theorem proving for example) are intractable. Another is that no one model of anything is good enough for all tasks. Reasoning from these laws can provide *theoretical* grounding for computational theories of the central systems of mind.

3) Mental models tend to form mathematical lattices under the relation variously called subsumption or generalization. Your mind puts a lot of effort into maintaining this lattice because it has so many important properties. One of these is that the more abstract models provide a normalized decomposition of world-situations that greatly constrains the search for useful analogies.

4) I have been using these ideas in building a computational theory of routines, the frequently repeated and phenomenologically automatic rituals of which most of daily life is made. I describe this theory briefly.

Acknowledgements. Bob Ingria made the tape from which these notes are descended. Betty Dexter selflessly transcribed the tape. The editing is based on part by comments from Mike Brady, David Chapman, and Margaret Fleck. All responsibility is mine. I have not tried to eliminate overlaps between these notes and other things I've written.

I took the title from Fernand Braudel's inspiring book about everyday life in the Renaissance and its significance in the broader history of that time. Fernand Braudel and James Joyce are the central figures in modernity's peculiar concern with the details of everyday life.

Most everyone's house has a stack of waste newspapers. There's a whole institution to stacks of newspapers. Ours is under a PDP-1 scope, but otherwise it's just like everyone else's. You know how in the middle of the Sunday paper there's this worthless section of paper that you could just pull out and throw away if it didn't have the Parade Magazine hidden in it? Well, I had just thrown that section on our stack of newspapers when something stopped me. Once I turned and looked down at the stack, I knew how to start this talk.

You know how when you're trying to give cough syrup to a small child, you've got the spoon and you've got a bottle of cough syrup, and you pour it right up to the brim? Everybody pours it right up to the brim so that the only thing keeping it on the spoon is surface tension. Really bright. And you try to put it in the child's mouth and it spills all over the place.

This routine of trying to give cough syrup to a small child is an institution itself. Everybody's had to do it once or twice. This spilling is also an institution. This notion of a thing that commonly goes wrong in the course of a routine pattern of activity is what I'd call a *hassle*. It's the sort of thing you say *what a hassle* about in casual conversations. And it's one of the things I've been studying over the last year or so. What I saw in the stack of newspapers was an advertisement for a new brand of cough syrup in gel form that doesn't fall off the spoon.

When I saw this out of the corner of my eye, it took me about a quarter of a second to realize that using it to start this talk would let me introduce two central ideas. The first is the notions of a routine, and particularly the use of identifiable hassles in recognizing opportunities to modify a routine. The second is the amazing phenomenon whereby I knew this advertisement is the right way to start my talk having seen it for about a quarter of a second. I hear different numbers for how long it takes a signal to propagate through a neuron. Whatever it is, divide it into 250 milliseconds and something very interesting was happening in that many neurons deep in my brain. People toss out perfectly funny jokes in casual conversation in less time than that. That's really remarkable.

I've been studying the mundane activities that rush by in your stream of consciousness without calling much attention to themselves. That they go by in your mind so fast and that you do them so routinely tells us a lot about the structure of the mind. I have a thesis,

as yet undefined, well, certainly as yet unwritten, that I want to defend today. It comes in three parts.

The first part is that the structures we find in everyday life reflect structures in the mind. I hope everybody here finds this intuitive, because we've been acting like we believe this for so long now. People around here sometimes call those structures in life *constraints.* But what I mean by *structures* and *reflect* I haven't told you yet.

The second part is that the mind is not the way it is out of arbitrariness or accidents of evolution. Instead, we build our minds as a rational response to the impediments to everyday living posed by fundamental laws of computation. We can write those laws down, and I will. The joking way to put it is that I can derive all of psychology from four axioms. This talk is mythically in two parts. This first part concerns about the first two of these axioms.

The third part is that we can find these structures in everyday life, and that one good way to find them is introspection. All of the modern history of psychological methodology notwithstanding. There are lots of criticisms of introspection out there, but I think what they come down to is that we don't have a good observation vocabulary for introspective observation of what goes on in people's minds. I want you to come away with the idea that computation provides an observation vocabulary for introspection. Don't think of it as *you can see yourself working like as a computer.* Think of it this way: computation provides a vocabulary of immense power for describing structures and processes, and we can use that vocabulary to observe ourselves and give interpretations to thoughts that otherwise sail by unarticulated.

Among my inspirations for this research is a book by Jean-Paul Sartre called *Transcendence of the Ego*, which says that phenomenologically the workings of your mind are external to you. They are transcendent in the same sense as any process out in the world is external to you, not something that is immediately given to consciousness. That's a pretty routine matter of experience and becomes an even more routine matter once you've learned to name what's going on in your mind as it's going by. Another of my influences is David Marr, who started (but, I claim, didn't nearly finish) the very hard job of providing a proper computational construction of the competence performance distinction. People are beginning to get it down for purposes of studying the peripheral systems of mind, but it's a

much harder thing to get straight for the central systems of mind, the parts that do the thinking and the feeling.

My method is simple. The theory predicts that a certain class of events will take place in your life, that you'll think a certain way and will act a certain way, or in certain classes of ways. I sit awhile and reflect on the prediction, and I convince other people to reflect on it too. Then one day in the midst of some entirely mundane activity, like making breakfast, or driving to work, or picking up a coffee mug, or putting down a coffee mug, or opening your mail, or choosing a turnstile on the subway, or using Emacs, or any of those things you feel like you're doing automatically, I'll wake up and say, *what just happened is an instance of the theory.* I don't know *why* this happens, but it does. When it does, I stop and I write it down. So that by now I have fairly thick notebooks full of relentlessly mundane stories about making breakfast and driving to work. Some of them are very funny, but most of them are boring.

I've discovered that I can learn an amazing amount about how people make breakfast that they weren't even fully aware they knew. Whenever anyone asks me about my work, after I tell them all this high-falutin stuff about the structures in everyday life, I bring the conversation around to, say, making breakfast, making cereal, say. A lot of people start fidgeting when I say *for example a lot of people have little tricks for slicing bananas onto their cereal they've invented and never been able to tell anyone about.* The one third of the cereal eating population who has such a trick falls into two groups. There are the relatively uninhibited people who start uncontrollably pantomiming banana-slicing actions that they've invented and that they've never been able to tell anyone about their whole lives. The most common one is slicing the banana with your spoon. Most everyone who does this thinks they're the only one in the world. I used to slice the banana with my spoon after somebody told me about the idea. I thought it was a great invention until I discovered that I had to move my thumb or my little finger to some strange place on the spoon to get around the fact that all the spoons I have have these narrow little necks so when you try to cut the banana you have no way to keep the spoon from spinning in your hand. Spoons weren't meant to cut bananas with; knives were. I've gone back to using a knife. More controlled people manage to restrain themselves, and deliver me scholarly lectures on the optimal way to slice bananas, sometimes extending for half an hour. I've learned an awful lot about slicing bananas this way.

Next I want to explain some of the intuitions that this work is built on. There's not much new about them. I want to talk about two classes of intuitions, ones concerning *lattices* (a lattice being a partial order with least common superior and greatest common inferior operations), and arguments, as in the method of dialectical argumentation. Intuitions about lattices and arguments account for an awful lot of the thinking that goes on in everyday life.

Ideas about lattices explain most of the symbolic inductive learning literature in AI, although unfortunately most of that literature is not stated in terms of lattice theory. In particular, theories of analogy are about lattices. If I have situation A and situation B, and situations A and B are analogous, it's by virtue of them both being instances of some C; if something is true of A that would be useful to carry over to B, it can be abstracted and put on C, and brought down to B. This is more roundabout than the account that says that you use analogies by translating knowledge across structural mappings. You don't really see the purpose for explaining analogy such a complicated way until you've got 100,000 A's and 100,000 B's very widely separated from one another, each of them in extremely complicated situations. A typical analogy will carry over five elements from one situation to another, and figuring out which five are important is a tremendous search problem, among other things.

Here's an example. I often come home late at night after all the lights are out. We have roaches in our apartment that come back every once in a while. We cover the place with poison and traps. We've even tried cleaning the place up, but nothing stops them. So part of my routine when I come home at midnight and the lights are out is to use this propane torch to kill a few roaches before going to bed. The way to kill roaches, as everyone knows, is to reach up and turn on the light, whereupon they all head for cover. You've got about a second to kill them before they're all gone. There's a problem, though. The torch has a knob with has three settings, Clean, On and Off, and the flame only comes out when it's On. When it's dark you can't see where the knob is, and when you turn on the light to move the knob to On the roaches scatter. I discovered this at very great length, and I found myself standing in the kitchen in the middle of the night squinting at this knob trying to see where it was, so as to get it to On. One night I thought *this is silly.* Then it occurred to me the foolproof way to get the switch to On was to turn the switch all the way around to Off, then click it back one. It would then be guaranteed to be On, wherever it started. I was

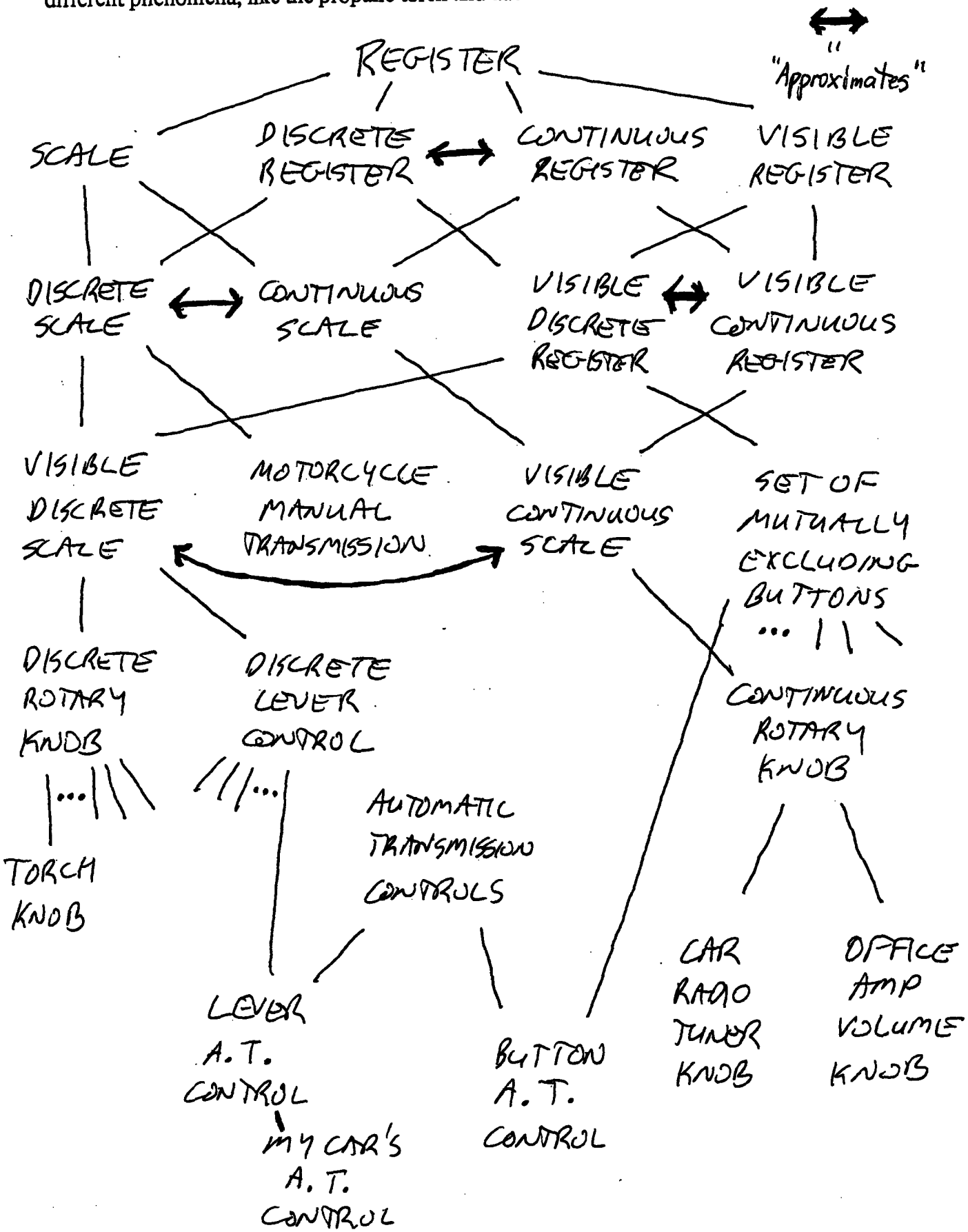
proud of myself. This is now part of my routine. I come home, the torch is always sitting in the same place, I pick it up, I turn it, I turn on the light and I blast some roaches.

Next morning after I discovered this trick I was trying to park a very large car in a very small space, in Chinatown. It had an automatic transmission, and I had to go from Drive to Reverse to Drive to Reverse to get the car wiggled into this little space with all these pedestrians walking in front and behind the car trying to get me to crush their legs. I didn't have any time to be looking at the transmission to put it into each new gear; my eyes had other things to do. A common pattern to changes in routines is freeing up your eyes to do other things, omitting perceptual steps by somehow forcing the world into a known state and then putting it from there into a state you want. (One place it occurs a lot is in Emacs, when the system load is high. Control-A puts the cursor at the beginning of the line no matter what and you can work from there. When the system finally gets around to redisplaying, you've gotten all kinds of work done if you were on the right line.) I remember the day when I first learned the order of the positions on an automatic transmission, long after I learned to use the automatic transmission. It's Park, Reverse, Neutral, Drive, Second, First, six positions. If you're in Drive, clunking it two left gets it into Reverse; if you're in Reverse, clunking it two right gets it into Drive. That much I had known for a long time, but I kept getting lost, so I'd still have to look and see where I was. The problem was I didn't have the thing in a known state. After about the fifth time this happened, right at that same point where I said *this is silly*, I made the analogy, picked up the lever, whacked it all the way left into Park and one right into Reverse.

The analogy between those two situations is based on a particular subset of facts from the two situations. These situations are characterized by 100,000 facts, of which about three are carried by a mapping to make that analogy. For purposes of analogy, a good representation normalizes your understanding of a situation so that you can make the proper analogenic generalization of a situation, from among the huge space of generalizations you might make. If you pull away the common elements of any good analogy, the resulting abstract model will have a certain hard-to-define sense of completeness or closure to it. It is worthwhile cataloging these models. I call them the *generic models.*

There are an awful lot of generic models. To get an idea of what they're like, have a look at Figure 1. This is a diagram of some of the lattice relationship among a variety of knobs,

Figure 1.
Here is a part of your mental lattice of models of knobs, buttons, and switches. One role of the generic models in the lattice is to mediate otherwise obscure analogies between very different phenomena, like the propane torch and automatic transmission of the text.



buttons and switches. The topmost elements of the lattice are the most abstract. These most abstract elements are Chapman's cognitive cliches. Cognitive cliches are the periodic table of representation, the simple models out of which more complex models are built. In a simple situation like this the models will tend to be just plain single cognitive cliches and more complex models will be more complex.

As you descend into the lattice, its elements become more and more specific. The bottom of the lattice is populated by the details of specific experiences from your life. A lattice is like a tree except that branches that spread out might rejoin further down. This is called *reconvergence.* The reason it has a name is that it's almost impossible to make a large number of algorithms execute proper reconvergences at the same time in a Connection Machine. This is a severe technical challenge for Connection Machine programming.

The analogy between the torch knob and the automatic transmission lever can be located by finding their least common generalization, which is the notion of a discrete scale. Some explanation will help. A register is a cell that has exactly one value taken from some set. A scale is a register whose set is totally ordered and has a pair of extrema. So for example, the range between 0 and 1, or the values Park, Reverse, Neutral, Drive, Second and First might be ranges that a scale could have. As you go down the lattice the models get more and more specific. Think of all of these models as being implemented by frames. Each link in the lattice specializes the idea above it to the idea below it. Each link either adds a new slot to the frame above, or adds some new restrictions on things that can fill existing slots.

My plan for using the torch knob depends only on the properties of that torch knob that are summed up in the notion of a discrete scale. There's a current position out of some discrete totally ordered set of values, I can tell when I'm at an extremum, and I have the option of moving either one left or one right at a time. The trick was to go all the way to the minimum and count from there to the place I wanted to go. Associated with every plan is an argument explaining why it ought to work. What's remarkable about my torch knob plan is that the premises of its argument refer only to properties of torches and knobs and roaches and fingers that can be abstracted in the idea of a discrete scale. I propose that I installed the plan and its associated argument in the discrete scale frame, hoping to apply it to some future discrete scale. The process by which an argument is associated with the most abstract frame that accounts for its premises is called *argument indexing. What facts about the current situation does this argument really depend on?* The more you can abstract

the premises of an argument away from particulars, the more you can generalize it. The average argument depends on fairly specific facts; one that doesn't is an important discovery. This is not a radical novelty; Paul Rosenbloom at CMU wrote a thesis about a fairly similar idea called *chunking.*

My plan for turning my torch on in the dark originated with the volume knob on the amplifier in my office. This knob floats between three different values, around 3 when I'm using the speakers during the day, around 5 when I'm using the speakers at night, and around 9 when I'm using the headphones. One day I pulled out the headphones and turned on the speakers, went away, came back after a while, and dropped the needle on the record. From that day on I have remembered to turn the volume back to a safe value before leaving the room. So part of my routine for leaving my office for any period is to turn the knob down to 3. Now I have better things to do with my eyes than look where the volume knob is, so one day I somehow invented the trick of turning the knob down all the way and turning it up to an amount that corresponds roughly to 3. The exact value doesn't much matter, but if I care about it I can get it somewhere between 2.5 and 3.5.

I certainly can't prove it to you, but I think that I came up with my torch knob trick by indexing the argument for the volume knob plan. When I abstracted away from it being volume and not temperature and the extrema being 0 and 10 and not 5.1 and 5.2 and the fact that I was leaving my office and a lot of other facts, I arrived at the notion of a continuous scale. A continuous scale is a pointer into some continuum, and you can engage in the process of moving it left or the process of moving it right, and you have a way of knowing whether it's at one of the extremes. Frequently you also have some rough correlation in mind between the degree of motion and other variables, like how much noise is going to come out.

I claim that I reformulated that plan from the continuous terms to the discrete terms. It works just as well to clunk, clunk, clunk as to go through a process. A smooth process is often reformulated as a series of steps and *vice versa.* This phenomena of reformulating between continuous and discrete models of things is absolutely ubiquitous in everyday reasoning. It's about the most common form of reformulation, changing of models for situations in everyday life. This is called *aggregation* by Dan Weld. Aggregation is reformulating from repetitious discrete events to a smooth process, and disaggregation is going the other way. As Chapman says, continuous scales approximate discrete scales. The

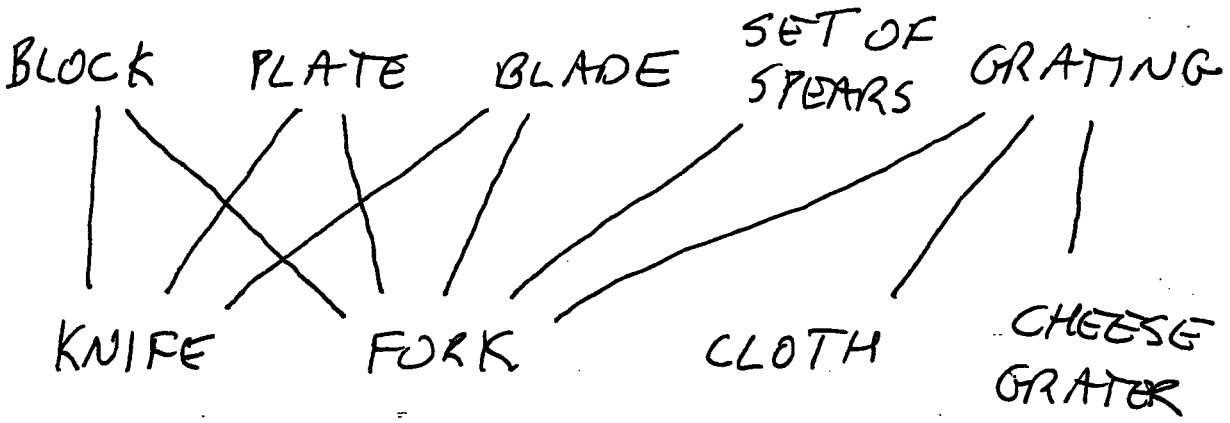
reformulation carries over all the ideas of each model, including your ability to move in either direction or to tell when you're at an extreme. So the argument for the continuous form of the trick can be translated into an argument for the discrete version.

The figure mentions some other examples. The only other one I want to talk about is the tuner knob in a lot of car radios, you turn it, and turn it, and turn it, and turn it, and you realize it doesn't stop at the end of the scale and you've been at 1600AM for the last five minutes driving down the road. You thought you had some idea how to fill in the slot that explains how to tell you're at an extreme, when in fact you didn't. So you formulate a wish. *I wish I had some way to tell when I was at the end.* You have a lot of these wishes floating around in your head. I wish I had a way not to spill the cough syrup. I wish I had a way to make automatic door closers not knock the packages out of my hands. I wish for an awful lot of things, most of them very mundane. Most of your wishes you don't even know about. Each wish combs through everyday experience looking for some way to make itself come true.

I also remember the day I discovered this phenomenon, noticing myself spinning the dial. The trick I was using to be sure I was at the end was to turn it the distance roughly corresponding to the whole length of the dial, so I'd be sure it was at the end. Doing so, I noticed a slight change in the amount of resistance the knob was putting up. The wish about a way to fill the slot explaining how to tell when I'm at the end of the scale raised its hand. I then checked whether that change of resistance meant I was at the end of the dial, and it did, whereupon that slot got filled back in again.

Let's have a look at this fork. I've been spending a lot of time looking at tools of all kinds lately, especially carpentry tools. Mike Brady, Jon Connell, Dave Braunegg, and I will someday finish a paper on tools called *The Mechanic's Mate.* I just want to mention a couple of the most relevant things from it. A good tool makes a simple model of its use work reliably. So a good knife is sharp and a good pan distributes heat evenly. Forks have about four or five ways they are meant to be used, and very many ways they *can* be used. And for each of those different ways of using a fork, there is a different best model of the fork. For stabbing things the fork is a set of spears; for scooping things up a fork is a flat plate; for cutting things with the side of the fork the fork is a blade; for mashing things it's a grating; for pushing things around it's just a block. And there are others. Figure 2 shows part of a lattice, with the general notion of a fork under all its possible models.

Figure 2.
Associated with each of uses of a fork is a different best model to adopt. The outlines of each plan for using a fork are provided by the generic plans associated with the relevant generic model. Thus, one way to wash out a grating is to run water through it from the back, whether the grating is a fork, a piece of cloth, or a cheese grater.



The ideas of blade, plate, grating, set-of-spears, and block are all generic models. Generic models are frames in your lattice, trying to apply themselves wherever they can be applied. Each model contains a stock of wisdom about what you can do if you are close enough to be correct in viewing something as being an instance of one of that model. A generic model, like any good representation, suppresses irrelevant details. When you're viewing a fork as a flat plate you ignore all the space between the tines; sometimes that's good enough. The grating model ignores the sharp edges; sometimes that works too. The rule is to plan with the most convenient model and to debug or plan defensively with the least convenient model. Just about everything that can go wrong in the course of using a fork comes from the fork behaving according to a different model than the one you were using. So if you try to use a fork as a flat plate and scoop something up with it, a couple of things might happen: it might be behaving as a block and pushing the peas ahead of it, or as a set of spears stabbing some of the peas and pushing the rest ahead of them. Now if you try to cut something it might be dull, behaving as a block. If you try to mash something, the tines might stab it and get stuck. When the fork bends or breaks or melts it's behaving as a plastic or a liquid (or whatever, according to your particular ontology).

Each these generic model has associated with it a number of generic plans. Blades have a plan for cutting associated with them. It says *cause the blade to pass through the thing along the plane you want to make the cut.* Plates have plans for mashing; blocks have plans for smashing or for scooping up; spears have plans for stabbing. To wash out a grating, turn it upside down and run water on it. That plan works for all the other things that are gratings as well, like cloth or cheese graters.

That's all I want to say about lattices and arguments. What I want to do next is frame all what I just said in terms of these long-awaited laws of computation. I tend to state what I call the Laws of Computation in somewhat facetious ways to indicate that this isn't yet, as I wish it were, a formal research program. Maybe someday. The first law of computation is that *no model is ever good enough*, because the world is a complicated and confusing and messy place, and because of the poverty of stimulus in any delineated domain. This law has a lot of corollaries. One that you can observe in kitchens and garages and other institutional work places is the contract that things behave according to simple models. You try to keep your knives sharp, and you try to keep salt in stock.

Another less visible corollary of the first law is that the mind spends an awful lot of time keeping track of models, trying to invent new ones, trying to apply old ones in new ways, trying to figure out which one to use when the first one it chose doesn't work. That's what the idea of a lattice of models is all about. The more abstract generic models at the top of the lattice can be applied in a lot of different situations, where the generic models at the bottom are more specific.The relations that hold among the models, the generalization relations and approximation relations, and probably all kinds of relations I haven't shown you yet, are the ways you move around in the space of models.

Part of making theories of these things is understanding what the generic models are in different parts of life and watching myself, for example, reformulating among them. The trash can in the men's room on the seventh floor (I assume the women's room is somewhat analogous) is set in the wall and has a narrow flat rim. I happened to set my coffee mug down on that rim and right before my eyes, the rim alternated between the model of a surface and the model of a blade. When you put something down on a surface it stays there, and when you put something down on a blade, it rocks a bit and falls into the trash can. One of the things that's on the approximation link between a flat surface and a blade is a suggestion to look for the rocking motion that indicates that it's behaving the wrong

way. The link also has a plan for testing if it's going to behave the wrong way, which is to rock it a bit deliberately and see if it cooperates, and a description of the usual way you knock something off an almost-blade, which is that your hand or some other tool knocks it while moving back toward you in the course of doing something else. All of which happened.

The facetious way of stating the other law of computation I want to mention right now is that *all problems in AI are search problems and all search problems are intractable.* (All exceptions granted.) For present purposes, it's a lot harder to prove a theorem than to check a proof. (Theorem proving is often formulated as a search in proof space.) If we really worked that way, we wouldn't be very smart, but the point stands nevertheless. So think of looking for a proof as looking for a path through a dense forest.

One of the corollaries of it being so very hard to prove a theorem, to find a path through the woods, is that when you've found a path it's a good idea to hang on to it. In other words, you spend a lot of time keeping track of old thoughts and trying to apply them in new situations because that's so much easier than having to face each situation afresh. In other words, just about all the thoughts you think are old thoughts. *That people tend to think and speak in cliches is an important constraint on the structure of the mind.* Argument indexing is a method for making maps of useful paths through the woods and applying those maps to as many new problems as possible.

Another corollary of the second law is in the notion of wishing. A wish says, I can't find a way through the woods, but I'm going to assign someone the task of looking out for a way through the woods from this point, in case I need to do it again myself. So in the back of your mind a lot of little mechanisms are looking for inspiration about ways to finish a search through the woods.

Those intuitions took several months to invade my daily life enough that I could actually see instances of them happening, because they're so very abstract. It's hard to recognize the applicability of anything that's so very abstract. More generally, things that are lower on the lattice are much easier to notice instances of than things higher in the lattice. There's a trade-off: abstract ideas are more widely applicable and concrete ideas are easier to use.

Now I want to tell you about the main thing I did once I had figured these things out for

myself. I had a long look at routines, such as administering cough syrup and driving to work and making breakfast. Routines are frequently repeated and phenomenologically automatic patterns of activity. Most of daily life is made up of routines. Routines are hard to study because they're invisible. You don't notice routine thoughts as they're going by in your head. People talk about themselves as if they're carrying out routines automatically, like the person who responded to the news that I (then) intended to make robot make breakfast by saying *oh, I'm a robot when I make breakfast.* People always sound a little nervous when they say things like that because they know the words *automatically, mechanically* don't nearly capture what it feels like to do these things routinely. My theory of routines is both a computational theory of what mechanical things are going on in your mind when you're doing something routinely, and also a phenomenological theory of what it's *like* to do something routinely. The interlinking of these two parts of the theory is important: it's hard to give articulation to what it's like unless you have a good vocabulary to describe it. And computation provides that vocabulary.

My theory of routines had its origin over the last summer, a time when my life was constantly changing around. I worked in different places and I lived in different places and I had different cars and I was on different schedules, and so on. Everything would switch around and my life would settle down to a new set of daily routines. The routine of getting lunch differs according to where I'm working and what my means of transportation are and so on. The routes I walk from different places change when I have different places to walk. Each time life settled into a routine, I wrote down what was happening that constituted settling into a routine. Observation led to theory led to observation led to theory led me to talk to you here.

Everyday activity is marked by improvisation and the gradual precipitation of routine patterns of activity. The first time you face a novel problem, one for which indexed arguments don't suffice, you have to muddle through. You solve it by rough analogies but mostly by weak methods. You use simple, safe things that work, like taking the interstates even though the back roads would do and using GPS even though there are canned algorithms that will do it in less than exponential time. The period of improvisation before a routine is set up is marked most particularly by the attitude you take toward the world. You're dealing with the world through simple representations, and the situations the world presents you are very often unanticipated. Your attitude is, in this sense, reactionary.

In the course of carrying out a routine you come to understand what the process of carrying out the routine amounts to. You make a model of the process of carrying out the routine. This model, like all models, is made out of frames in this lattice. If you're lucky, the model will contain indexed arguments about how better to carry out that activity. This is called *internalizing* the process.

As a routine develops, it takes account of finer details on the part of the world you're moving through. It does so by using successively different models, making successively smaller changes, noticing successively smaller things. Settling down a routine is a skill for each particular domain. There are particular skills for settling down a routine for walking from here to there, for carrying out a recipe, for using a tool. There are a lot of particular skills for making a routine settle down. I have a rough theory about how this settling down works. It says that routines develop in stages. (A more accurate statement would be that any particular domain the skill of settling down a routine develops in stages.)

I'm going to distinguish five stages.

The first stage is the stage of recapitulated reasoning; its mechanism is simple argument indexing. Your mind takes arguments that worked and stores them away for future application. If some strategy works fine the first time, then you'll keep on doing it that way. So if you drive somewhere and the route it doesn't seem to be taking any longer than it ought and nothing especially dangerous or memorable happens and nobody tells you about a better route, then without even thinking about it you'll use that route again, unless you're the kind of person who always enjoys finding the very best route.

The second stage is the primitive credit assignment stage. My skills at the development of routines for navigation are stunted at this stage, I've decided. My evidence is that I have many pairs of places that I walk between routinely, and my routes are often not reverses of one another. One example is walking between Technology Square and the Institute main campus and back, something I do an awful lot. The route I walk from my home to the Washington Street subway is very different in the morning from the route coming back in the evening. They're roughly the same length but it never occurred to me to even think that they were different.

Here's why. Phenomenologically speaking, in this second stage you're still reacting to the

world to a fair degree, behaving myopically. As I walk, I make each choice about which way to turn based on considerations that the environment immediately suggests to me. I have no map, or only a very sketchy one. Finding yourself at a choice point when walking somewhere is a metaphor for all decision at this stage. Often you have a decision to make and you can't think of any argument that distinguish between the two choices, you can think of arguments both ways, but you can't think of any way to distinguish between the arguments. (An example is when you're sitting in a restaurant staring at a menu full of perfectly good things to eat for half an hour, unable to decide exactly which one to eat.) At those decision points it frequently in a very faint way occurs to you to try and come up with some more evidence to decide which way to go, but the evidence really has to impose itself on you. You've got to see the puddle that you had to walk through when you went this way last time. You have to see the bushes you had to squeeze through that way last time. Out of sight, out of mind.

So, just the other side of Main Street are buildings 45 and 48. You walk between Technology Square over the railroad tracks over to the main campus, you go through this parking lot and there's this sidewalk with hedges along it. Does everybody know the place I'm talking about? You walk through a parking lot there. Now imagine coming the other way. There are three routes to go. One is down that sidewalk. The problem with that sidewalk is there's this hedge to your right and you end up about 10 yards further along Main Street than you'd like to be. And a second route is along that rutted dirt road whose ruts always have six inches of water in them. It's not very much fun to walk along when it's dark. The third on is to cut between the bushes along the path everyone's worn into the grass. They plant new bushes and lay down new the grass about every six months, and it's worn down in two or three days. Through a series of applications of primitive credit assignment, I've evolved this complicated way of deciding which of those three paths to take, depending on how recently it's rained, whether they've put in bushes recently, how the cars are arranged in the parking lot, and how dark it is. (And I'll bet you have too, and aren't willing to admit it.) But even so, I take an entirely different route when going the other way.

The third stage is called scene characterization. Routine evolution, as I have mentioned, involves the internalization of process, and the first process representation is very much like Schank's idea of a script. Scripts are made up of scenes. Once I was writing a paper about

this over at the late Atari Cambridge Research. Every half an hour I'd get up and make another cup of tea. I hadn't used this kitchen and its particular set of tea-making technology before. I'd go in, I'd get the teabag and put it in the cup, I'd get out the kettle and fill it with water, and I'd put it on the stove. And one day while I was doing all this it occurred to me that I should be putting the water on the stove first and getting the other things out afterward. The next time I went into make tea, I went straight for the kettle, filled it, and put it on the stove. But then I thought *hang on, this is an electric stove.* The next time out I turned on the electric stove before filling the kettle.

This story illustrates two instances of a very common way of rearranging a routine, moving some activity into a waiting period. An especially complicated instance is the subway commuter's trick of standing on the platform so that when you get in the subway and out at your destination station you'll be standing in front of the exit. There's a red and white sign hanging from the ceiling on the Washington Street northbound Red Line platform; if I get on the train right by this sign, I'll be right in front of the most convenient exit in the Kendall Square station. You're going to have to walk some distance along train platforms no matter what; with this trick you do that walking while waiting for the train rather than after it arrives.

Suppose you didn't have the idea of *waiting* in your head. You could spend ten minutes every morning of your life staring into space until the water boils without ever clearly formulating what exactly is bothering you. But, fortunately, waiting for water to boil and waiting for an electric stove to heat up are notorious hassles. By putting names to hassles, you provide yourself with a way of noticing that something is more trouble than it ought to be. (There are other heuristics for noticing excessive effort. It's just not trivial.) There's a lattice of hassles. (Of course; there's a lattice of everything.) Near the bottom of this lattice are hassles peculiar to particular routines, like spilling cough syrup or bagels getting stuck in the toaster. Higher up in this lattice are the *generic hassles*, like waiting, spilling things, having to work in a clumsy position, bad vibes from any other person, and any number of others.

Associated with each generic hassle in the lattice are suggestions about what to do about it. So one way to avoid waiting is to try to move some activity into the waiting periods from somewhere else. You can avoid making a mess by spreading newspaper to catch it. You can avoid spilling something by using a larger container, making several smaller trips, or

increasing the stuff's viscosity. I submit that it's not obvious *a priori* that you can alleviate the hassle of waiting by moving useful activity into the waiting period. If you didn't categorize hassles, it would require a fresh miracle to figure this out on every new occasion of waiting. In the Mechanic's Mate paper, we give a list of generic hassles that come up frequently in carpentry, like having to work in tight spaces or always having to alternate tools. Generic hassles suggest generic repairs.

The fourth stage of routine development is the stage of process characterization. Now I have to stop being sloppy in distinguishing between a routine and a plan. A routine is not a plan. Recall how I spoke of routines as precipitating from improvised activity. The more your improvisation responds to discovered contingencies of the environment instead of imposing a preconstructed plan, the more likely there is to be unanticipated structure to the activity. As the improvised activity settles into a routine, you can notice that structure and use it in changing how you carry out the activity.

My favorite example (made up, though the pattern is common enough) is vacuuming the dining room. Vacuum. Vacuum. Vacuum. As I vacuum up to this chair, the prerequisite for making further forward progress is defeated. So I move the chair and vacuum under it. Very good. And I vacuum along and there's another chair, and I move that one out of the way and I vacuum under it and I move along and there's another chair. Oops, I move that out of the way, and I vacuum under it. Slowly it dawns on me that there's a pattern. I'm moving a lot of chairs. After noticing that, I can build a representation of the vacuuming process as one of interleaving the processes of floor-vacuuming and chair-moving. There are generic processes, and the idea of interleaving is one of them. The generic process of interleaving suggests you try deinterleaving. Move all the furniture, then vacuum the whole floor.

These later stages are marked by your global understanding of the process of carrying out a routine. Where once you were reacting to environmental contingencies, now you are participating in a process. Imagine you're riding a roller coaster. It makes a big difference to the experience whether you understand what a roller coaster is.

The final stage of routine development is called process manipulation, a much more formal version of the previous stage. The prototype of this stage is not in everyday life but in business management. Operations researchers and strategic analysts make elaborate

models of parts of corporations and apply abstract manipulations to them to come up with revised administrative procedures. Or that's the theory. They call it "rationalization", which I find an interesting choice of name. The word routinization, in fact, I got from Alfred Chandler's work on the development of modern corporate organization.

I want to finish by sketching where this is going. Yes, there's going to an implementation. I told you that this was the first part of a mythical two-part talk. The other part of the talk is about the other laws of computation. What follows from the other laws is what we know as the theory of personality. These laws of computation are equally facetious, or more so, at this point and I don't understand them as well. The first is something like *everything that happens needs an explanation.* Several years ago someone kidnapped a whole bus of school children. After they were ransomed, some psychiatrists asked these kids why this incomprehensible thing had happened. The kids all had bizarre explanations of things they had done to cause this bad thing to happen to them, things like stealing cookies, or going into the living room when mom was away and she told them not to. Because it's important to have explanations of things that happen, the dynamics of trying to explain things that happen explain a lot of the dynamics of what goes on affectively in one's mind.

The last one I understand even less. I've been concerned throughout with very mundane thoughts that sail by without your paying any attention, and so are for any practical purposes unconscious. Consciousness and unconsciousness are matters of what thoughts you pay attention to. You can fail to pay attention to a thought either because it's so mundane and unproblematic that you never had any reason to pay attention to it, or because you're actively repressing it. The law says, *attention is a resource.* One reason you have a self is to focus your attention on things that are important. Your personality is in large measure your strategies for focusing your attention on different levels. You know that look people get on their faces when they're doing something dumb, and down deep they know it, but they'd rather not notice. That sort of not really seeing what's going around them. That phenomenon is continuous with people not noticing the thoughts they're carrying out as they make breakfast. Both are policies about attention.