From Humans to Human-level AI

Sean Markan

March 17, 2017

Abstract

This report discusses the problem of human-level AI, presenting an opinionated view of what the central difficulties are and what strategies and methodologies are likely to help. I suggest that the central problems involve modeling the richness of human thought and the acquisition of symbolic knowledge, and that important strategic decisions include modeling humans, seeking a "baby machine," and using "sensorimotor prostheses." I propose a methodology consisting of observation and reverse-engineering of human thought, in the context of reasonably challenging cognitive tasks. This methodology is illustrated through a variety of investigations into representations, concepts, and learning.

Contents

1 Preface

What would it take to make a machine that thinks like a person? Scientists have been pondering this question since the birth of AI in the 1950s. But some 60 years later, progress has been disappointing. We have many impressive specialist AI systems (like self-driving cars), but where are the machines that can pick up *any* new task without being reprogrammed, or understand a book on *any* subject, the way humans can? Where are the AIs that can *think*? Unfortunately, not only has progress towards true thinking machines been slow, AI research has become divorced from that objective, focusing instead on less ambitious goals which are unlikely to help with building thinking machines.

There are two popular schools of thought on human-level AI (HLAI). Each has a substantial following (both within and outside the research community), and each is flawed. The first school of thought is that current AI research (e.g. deep learning) has finally found its footing on the path to HLAI. This is far too optimistic; as we'll discuss soon, today's trendy AI techniques do not address the core issues of HLAI any more than did the trendy techniques of 10, 20, or 30 years ago. The second school of thought is that HLAI is a problem for which we have no attack[1]—so all we can do for now is plod along and hope that the situation changes one day. Although this second school of thought rightly recognizes the disconnect between HLAI and current trends in AI, I think it is too pessimistic: I believe there actually *is* a research strategy that provides a plausible course to HLAI, more so than the majority of current AI research.

This report is an attempt to explain that strategy. The strategy will be an unconventional one, setting aside many of AI's current preoccupations—including neural networks, many aspects of perception and motor control, and most of statistical machine learning. Instead it will lean on ideas further from the zeitgeist, such as cognitive modeling, introspective reports of cognition, and symbol manipulation. (And before you say "dustbin of history," let me reassure you that there will be new twists on these old ideas.) I hope you'll consider the upcoming proposals with an open mind—they may be unpopular but it's time to try unpopular things!

Who this report is for: I hope it will be of interest to anyone curious about the future prospects of human-level AI. The technical parts do (unavoidably) assume a technical background, but the first several sections should be understandable to anyone.

Finally, since HLAI is inherently a multidisciplinary topic, I hope that experts in each of the various areas I touch upon (especially psychology and linguistics) will forgive my incomplete knowledge of their specialties. This is an evolving document and I would be happy to be corrected on my oversights.

[1]Richard Hamming discussed this idea of "having an attack" on a research problem in his well-known talk "You and Your Research"—see Kaiser (2010) for a transcript.

2 Introduction

First of all, what *is* HLAI? I take the term HLAI to mean a computer that is as smart as a human—one that can do all the same intellectual tasks a person can, from summarizing a document to controlling a robot. It should be able to do all these things to the same standard of quality as a human, and it should not need any special programming or training beyond the sort of education ordinarily given to humans.

Note that I do *not* have in mind (nor do I think we should aspire to build) AIs that are "just like us." I don't see a reason to build machines which get angry or jealous, or feel pain (whatever that would mean), or have their own self-chosen goals and inclinations. I envision machines that are quite like today's computers, just not so dense.

The consensus among AI researchers is that HLAI is possible in principle. However, we are not particularly close to achieving it. There is a variety of opinion about just how long it will take, but you would be hard pressed to find someone confident it will happen in 20 years or confident it won't happen in 100. Of course, the harder we work on it, the sooner we will succeed.

The aim of this report is to explore how HLAI might plausibly be achieved. The report has two parts. In the first part, I will address this question in the abstract, with the goal being to orient you to the problem and to defend a particular set of research strategies, principles, and methods that I think should be pursued. In the second part of the report I will show these ideas in action, by means of several concrete case studies. These sections should put some meat on the bones developed in the first part, and hopefully convince you that I am proposing something specific and actionable.

Sticking for now to the first issue (how, broadly speaking, could we build a HLAI), I have tried to organize the discussion according to several "perspectives" which I think cover the main issues:

- The "strategic" perspective: what is the overarching strategy by which an HLAI could be constructed?
- The "historical" perspective: where do we stand now and what can we learn from the past?
- The "cognitive architecture" perspective: what would an artificial mind look like structurally?
- The "missing ingredients" perspective: what capabilities are needed for HLAI but have been missing from past and present AI systems? (I will argue that these mainly concern what we might call "thought" (or reasoning, or semantics) and the acquisition of symbolic knowledge through experience.)
- The "methodological" perspective: in light of all of the above sources of guidance, what concrete research directions are likely to be most fruitful?

The plan is to start by briefly touching upon the history of AI, in order to provide context for everything that follows. I will then spend some time on the question of "missing ingredients" in AI. I hope this will persuade you (if you aren't already) that we really do need to do something genuinely different if we ever want to build HLAI. Additionally, I hope it will give you some sense of *why* HLAI is such a hard problem and what demands a plausible strategy needs to satisfy. Next will be a discussion of general strategy for HLAI, which will lead us into the topic of cognitive architecture. That will then allow me to lay out what I currently believe is the most promising research methodology.

The rest of the report will illustrate that methodology by developing a variety of cognitive models, ranging from rather informal to highly formal. These will address questions about representation of thoughts and knowledge, the nature of concepts, how reasoning interacts with behavior, and how a HLAI could learn about categories through deliberate investigation. (This second part of the report will be overviewed in more detail in §8.)

3 A brief history of (HL)AI

Needless to say, human-level AI is a hard problem. In the 1950s and 1960s, when AI research was just getting started, many researchers were optimistic that they would be able to build machines with all the capabilities of human minds in a matter of years. Those hopes were not borne out, and researchers realized over time that many human abilities are extremely difficult to replicate. Human reasoning is subtle and complex. Humans possess vast amounts of what is called "commonsense knowledge" (e.g. "objects move in continuous paths") which is difficult to convey to computers. Human systems for motor control and perception are very complicated. Many other issues crop up as one dives deeper into the problem.

As a result of early failures to build anything resembling human-level intelligence, by the late 1980s the field of AI had moved away from that goal and toward new goals. AI became the art and science of doing tasks that humans do well but which are difficult to automate. The goal was no longer to build machines that could do *everything* humans can, but rather to build systems to do one particular task, or work within one particular conceptual domain, such as recognizing faces or controlling a particular stage of an assembly line. This new orientation met with much success (in part because of increases in computing power and the adoption of statistical machine learning), and it continues to be the dominant paradigm for research in AI to this day. But unfortunately, the "one task at a time, easiest first" paradigm is not particularly helpful for HLAI. Certain types of tasks are susceptible to this sort of approach, but many other types of tasks and capabilities (including anything that requires "common sense" or "genuine understanding"[2]) are not—and it is the latter class of capabilities that is crucial for HLAI. In fact, the sorts of problems that now dominate AI research aren't even "laying the foundation" for HLAI—they are just things a HLAI could figure out for itself if we solved the core issues, as we will discuss in more detail in §6.1.4.

In light of the above, it's fair to say that the goal of a *generally* intelligent system which could adapt itself to any new task (as humans can) has not been a serious focus of AI research since the 1980s.[3] You might be wondering whether any other field has "picked up the slack" in the quest to build thinking machines: what about cognitive psychology, neuroscience, or linguistics? The answer here is "basically no." Although each of these disciplines has something important to say about HLAI, each has methodologies and objectives that are quite far from "build thinking machines." So to a first approximation, currently next to no one is actually working on HLAI.

Given the lack of effort and interest in HLAI over the past few decades, one might wonder whether the problem is intrinsically unsolvable, or unsolvable at least for the foreseeable future. Personally I don't buy that line of reasoning. For one thing, long stretches of slow progress obviously do not imply there is no progress to be made—in the Middle Ages, 1000 years went by with hardly any scientific progress. Also, several other factors more plausibly explain the lull in HLAI research, including

- early disillusionment with the difficulty of the problem;
- the cooling (warming?) of the Cold War and the associated reduction in funding for basic research;
- the presence of a great deal of unrelated low-hanging fruit in AI (and computing more generally), which consumed attention that might otherwise have gone to HLAI; and
- the fact that HLAI will probably require a mixture of attitudes and methodologies from multiple disciplines (AI, psychology, linguistics) and academic incentives have evolved to make multidisciplinary work difficult.

Another factor not to be underestimated is simply *fashion*, which plays a considerable role in determining what topics receive attention from scientists. In the early 1900s, the dominant paradigm of psychology (behaviorism) discouraged investigations into thought, reasoning, and language, a position that seems crazy now. Even today, in linguistics the "generative grammar" paradigm is the dominant one despite a number of (in my opinion) more promising competitors.[4] In a similar way, human-level AI is out of fashion (disreputable, even) in academic circles. Part of the received wisdom in AI is that we are "stuck" on HLAI, and this has become a self-fulfilling prophesy, even though most AI researchers believe HLAI is possible in principle.

The bottom line then is that even though we aren't making much progress on HLAI currently, there is no good reason to think we *can't.* At the same time though, because professional researchers by and large do

[2]What is genuine understanding? One diagnostic would be the ability to carry on a natural conversation about whatever domain one is working within.

[3]Some readers may be aware of recent work on so-called DQNs, which have been used for example to learn to play Atari games Mnih et al. (2015). This work is sometimes cited as an example of modern AI research making progress towards general intelligence. However, the sort of generality displayed by DQNs is not the same sort of generality humans exhibit. DQNs don't do anything resembling thought or reasoning, and they require too much experience (far more than a human) to become competent at their tasks. As we will discuss, these two issues (thinking and efficient acquisition of knowledge) are probably the most important obstacles we need to surmount to achieve HLAI.

[4]A discussion of these issues can be found in Culicover and Jackendoff (2005).

not engage with the topic of HLAI, there is not much in the way of agreed-upon conceptual scaffolding for thinking about the problem. For example, what exactly is it that we need to figure out? In general terms what would a HLAI look like? What research strategies and methodologies make sense? We will address all these questions in the next few sections.

4 What's missing in AI

For the many tasks we cannot currently automate, generally the obstacle is that either (a) we cannot yet build a system that interfaces with the world in the ways the task demands or (b) the task demands reasoning that is too sophisticated or involves too much knowledge (or it requires new knowledge to be learned on the fly). As an example of the first type of obstacle, the difficulty of perceiving and manipulating things in an uncontrolled environment makes domestic robots hard to build. As an example of the second, researching a topic and writing a report on it requires far more knowledge and reasoning skill than we can currently get into a machine.

As I will argue in more detail in §6.1.4, I believe the problem of getting an HLAI to interface with its environment (i.e. through vision and robotic manipulators) is one we can bypass (by letting the HLAI solve it). The remaining problems then principally concern knowledge and reasoning. Since there has actually been a great deal of work in those areas throughout the history of AI, I want to focus here on the aspects of the knowledge/reasoning problem that we *don't* understand very well. In my view there are two prominent problems:

1. How can we replicate in software the kind of richness and sophistication seen in human thinking?

2. How can we acquire (symbolic) knowledge from experience?

I call these the "thinking gap" and the "learning gap." Let me discuss learning first, since I will have more to say about the richness of thought. (That's not to say that learning is any less important, only that it's a bit easier to summarize: the question is "how is X learned," for every value of X!)

I want to emphasize right off the bat that I am talking mainly about acquiring *qualitative* (or symbolic, in the sense of §6.2.1) rather than quantitative knowledge. When the sort of knowledge to be acquired can be described as a bunch of numerical parameters, there are already a great deal of established algorithms for this (this is what machine learning is good at). But when it comes to acquiring symbolic knowledge, very little is understood.

For example, learning that "faucets usually spurt out water when you rotate their handles" is the sort of problem that has received relatively little attention. Consider what makes this nontrivial. It's not just a matter of recording how many times you've rotated a faucet's handle and in what percentage of them water came out. The problem is much deeper. How does the agent properly frame the action it took (or observed being taken) as "the turning of a handle" rather than "rotation of left hand by 88°"[5]? How did it know to even try applying force to the handle? How did it decide to conceptualize of the ensuing outcome (which after all, would be just a sequence of video frames and audio samples) as "water spurting"? How did it come to have a "faucet" concept or a "water" concept? Of course, in this "faucet" problem, these questions are all complicated by the difficulties of processing sensory input and controlling motor output, but those complications can be stripped away by using artificially constructed environments, and there has been very little work that attempts to do so (notable exceptions being Drescher (1991) and Bergman (1995)).

In my view the difficulty of this sort of learning stems from two sources. First, one needs appropriate concepts and formalisms in terms of which the acquired knowledge can be represented. Second, one needs the wherewithal to *figure out* regularities in the world; this includes both an "investigative skillset" (exploring, forming and testing hypotheses, noticing patterns) and such cognitive capabilities as remembering statistics, grouping related experiences, and inventing new concepts. We are currently far from a solid understanding of any of these issues. My own take is that we should look at what humans seem to do, an approach I will illustrate in a number of models presented in the second part of this report.

Now on to the topic of thought. The general problem I'm getting at here is a broad one, and many similar or synonymous problems are inextricably linked to it: common sense, reasoning, semantics, knowledge

[5]Rotating the left hand by 88° is in general neither necessary nor sufficient to rotate a faucet handle.

representation, natural language understanding, the so-called "grounding problem." The issue centers around what sorts of symbolic representations ought to be used to represent information in the mind of a HLAI, how those representations get there, how old thoughts combine to yield new thoughts, and how thought leads to action. Now, a great deal of work has already been done on computer reasoning and knowledge representation. But it has not proved to be enough, and I think part of the culprit is that we have not sought to replicate *human* thought. Automated reasoning systems use representations and methods that look much different from—and much simpler than—what humans do. We will need to understand human thinking in much more detail to create HLAI. The next section will go into this issue in more depth.

5 The thinking gap

One of the clearest ways to see that something is missing in modern AI is the fact that—as far as I know—there are very few systems that have anything resembling a human *train of thought*.[6] There do exist numerous systems which can perform logical deduction, but that is only a small piece of human thinking, and in any case most of these systems do it using techniques (such as resolution) that do not particularly resemble those used by humans. I call this gap between what current AI can do and what humans can do the "thinking gap."

Now, the absence of systems with "trains of thought" does not necessarily tell us that getting machines to have them is the sticking point in HLAI. It could be (and I think this is true to a large extent) that AI researchers mostly haven't had occasion to build such systems because it is hard to bring "human-like trains of thought" to bear on the sort of practical applications they are primarily concerned with (at least until we learn how to get those trains of thought to be quite sophisticated—note the chicken-and-egg problem there). Nonetheless, the fact that today's AI systems don't have any process resembling human thinking should at least tell us that something fundamental is missing in modern AI research, and we need to remedy that if we are to achieve HLAI. But what specifically is the "special sauce" that human thought has and computers so far don't? We will address that next.

5.1 Thinking is complicated

To get a sense of the scope of the problem, let's take a quick tour of human thought. Let's begin with some hypothetical snippets of the inner monologue that plays out in each of our heads all the time. Suppose you've been asked to write an essay on the likely consequences of a government-mandated 35-hour workweek. Perhaps your thoughts run like this:

> Well many people work a lot more than that, so one might be concerned a lot less would get done. Even if theoretically people don't get much more work done in 50 hours than 35 due to fatigue, having 50 hours of availability makes scheduling and coordination easier. On the other hand, France has (or had) a policy along these lines without serious damage to its economy, and it would be beneficial in terms of work-life balance. But one also wonders how enforceable such a policy would be; it's not like the hours of white-collar workers are auditable by the government, especially when they work at home after hours...

Or suppose you need to make plans for an upcoming trip to Las Vegas for an AI conference.

> The conference a month away so I should probably book things soon. Do I want to stay longer than the conference? I don't have much interest in casinos but my sister Jane lives in Los Angeles—is that close enough to be worth it? Maybe I'll call her and check on her plans. It's too early in her time zone now though, let me write that on my todo list for this afternoon. The other thing is the hotel; let me look at the email about conference discounts...

There are a few things to note here, each one of which poses a major challenge and represents a way in which current AI systems fall short of human thinking.

[6] One exception that comes to mind is Ganesalingam and Gowers (2013), which tellingly came from researchers focused more on mathematics and linguistics than on AI!

- **The content of thought.** We can and do think about all sorts of things, and one topic can very easily call upon knowledge of something completely different (note how audits and casinos came up a moment ago). There is so much "common sense" knowledge a human knows that we can't hope to program it all into an HLAI; the AI will have to learn in the ways humans do: from experience, from reading, from being told, and so forth.

- **The texture of thought.** Thought does not in general proceed in a predictable, mechanical fashion. It is a mixture of goal-directed reasoning, recollection of related information that may or may not prove relevant, spontaneous idea generation (including not just ideas on how to accomplish something but ideas on what to accomplish), posing questions, drawing forward inferences (e.g. "I should probably book things soon"), decision-making, creation of hypotheticals, assumption-making, and many other things. This sort of texture applies not just when one's mind is wandering but in a number of contexts that might seem like they ought to be more organized, such as many problem-solving, planning, strategizing, and design tasks.

- **The expressiveness of thought.** Human thought involves a great amount of diversity in the types of thoughts we can have. This phenomenon deserves its own section.

5.2 The expressiveness of thought

Human thought is highly expressive, far more so than the formalisms we have so far constructed to model it. To appreciate the contrast between the types of reasoning we can formalize already and the full palette of human thinking, consider "first-order logic," the main representational language used in AI, linguistics, and mathematics (often with elaborations, but usually small ones). The types of propositions expressible in first-order logic are limited. First one can express that something is true of an entity, for example that Bob is bald:

$$\mathit{Bald}(\mathit{Bob})$$

(Keep in mind here that just because the computer can store this proposition doesn't mean it has any idea what *Bald* means or what *Bob* is; making it "understand" such things is discussed more in §5.3.) One can also express relationships among multiple entities, for example that Jane is the mother of Jack:

$$\mathit{Mother}(\mathit{Jane},\ \mathit{Jack})$$

Beyond that, one has a way to say "and," "or," "not," "every," and "some." For example, "every person has a father" can be rendered as in (1).[7,8]

$$\forall x.(\mathit{Person}(x) \Rightarrow \exists y.(\mathit{Person}(y) \wedge \mathit{Father}(y,x))) \tag{1}$$

Now consider some of the many ways in which human thought goes beyond the confines of what can be said in FOL (most of these are exhibited in our paragraphs above):

- **Vagueness.** Nearly everything we say has some element of vagueness. Sometimes it is relatively "overt" (as when I used the phrase "serious damage" earlier—how serious is serious?), and sometimes it is just lurking in the background where it normally doesn't affect things but nonetheless precludes making inferences with certainty. For example, is "I attended the meeting at noon today" false if you attended for only half of it? If not, the inference that a person who attends a meeting knows what happened during it cannot be made with certainty. And what if one wants to say something is "usually" true? How usual is usual?

- **Modality.** We can say that we "can," "must," "should," or "might" do something. While so-called "modal logics" provide one way of augmenting first-order logic with modality, they don't really capture the breadth or sophistication with which humans use modality. (This will be discussed more in §5.3.2.)

[7] This can be read "For every entity x, x being a person implies that there is some other person y who is the father of x.

[8] Actually, this simple proposition also illustrates another problem with first-order logic, or at least with taking its standard semantics too seriously: if we take (1) to be "exactly" correct, we arrive at the erroneous conclusion that there have existed an infinite number of people. Situations like this (which occur everywhere) mean that we need some better understanding of how to capture the qualifications and limitations lurking beneath our statements.

- **Individuation and abstraction.** We can view the same set of circumstances at multiple levels of granularity. To borrow an example from Pinker (2007), was the collapse of the Twin Towers on 9/11 one event or two? We can see it either way (or for that matter see all of 9/11 as one event). Similarly, we can wrap up complex ideas in what have been called "shell" nouns (Schmid, 2000) and think of them as units. And we can describe in simple terms a process that was much more complex—for example, saying "we used Excel to make that graph" conveys the gist of a much longer series of steps. Knowing the right degree of detail to use in such a case is a complicated pragmatic question, and unpacking the gist to whatever degree is necessary for a given purpose is even harder.

- **Sentence structure beyond subject-verb-object.** In addition to the basic subject-verb-object structure of sentences that is naturally captured in first-order logic, there are a great many other linguistic phenomena that appear in sentences (which are presumably a reflection of thoughts). These include tense, aspect, parentheticals, adverbials like "I go bowling <u>each Friday</u> <u>with my kids</u>" (which can be used to express a huge and heterogeneous variety of meanings), and so on. Really, anything we see in sentences presumably comes from some corresponding device in thought. And there is a lot of complexity in sentences: to get a sense of the scale of the problem, consider that *The Cambridge Grammar of the English Language* (Huddleston and Pullum, 2002) is almost 2000 pages, and is only an overview!

- **Analogizing.** In our thinking we detect analogies between the various things we think about. We might say, for example, "that's the same situation we were in earlier."

This list could go on for quite a while, but that should be enough for you to get the idea.

Now, I don't mean to imply that there are no ideas for how to formalize the sorts of constructs mentioned above. Quite the opposite is true; we have things like temporal logic, the situation and event calculi, modal logic, fuzzy logic, and so on. The problem is threefold. First, there are a lot of phenomena to cover, and the various ideas for each one haven't been integrated in a compelling overall framework (especially one which interfaces with the rest of cognition). Second, upon upon closer inspection, often the existing ideas don't seem "quite right" as models of human thought—they seem to capture part of the story, but not all of it. To give one small example, in modal logic one can construct downright strange things like "it is necessary that it is possible that X" (and then different modal logics disagree about what can be inferred from such oddities). Third, it's likely that there is a great deal of nuance in these areas that will only be exposed as we try to design thinking machines and let them loose on harder tasks than have been pursued so far. For any given formalism attested in the AI and linguistics literature, much elaboration and alteration will probably be needed to produce human-like trains of thought. So the bottom line is that there is still a lot of work to be done before we will have well-worked-out formalisms modeling the full range of thinkable thoughts with sufficient accuracy to allow for a functional HLAI.

5.3 The semantics gap

In addition to the problem just discussed (the *expressiveness* of thought), there is another substantial obstacle that HLAI must address: thought is also notoriously *slippery*. In other words, it's usually hard to precisely pin down exactly what any given sentence (and by extension any thought) *means*—that is, its *semantics*. By the way, when I talk about semantics, I have a purely operational definition in mind. A hypothetical HLAI needs to know when it's apt to characterize a situation with a certain word or sentence, and conversely what may be safely inferred from an utterance. The ability to do these things is all I ask from a theory of semantics (in contrast to many philosophers and linguists, who often seem to want some "deeper" account of meaning). But, as I will now illustrate with a few examples, wherever one looks one finds complexities that make it hard go get a system that properly understands and uses language.

5.3.1 The amorphous nature of concepts

Many scholars have pointed out that the meaning of a given lexical item is often quite amorphous. Wittgenstein gave the example of a "game" as a word which can apply in all manner of situations, many of which have little in common with the others. Another classic example is "climb", which can refer to ascending (when speaking of airplanes), to making a "climbing motion" (as in climbing down a ladder), or most typically to both. And as we talked about earlier, every concept is fuzzy at the boundary: how much of a gap can exist before a window can be said to be closed? How many is a crowd?

Even when we have a way to operationalize a concept formally, there is usually a big gap to connect it to actual usage. For example, a mathematician might define a curve as a "continuous map from $[0, 1]$ to $\mathbb{R}^2$," but that definition cannot be *directly* applied to understand an utterance like "this curve here in this image" because the actual curve is not perfectly thin, can only be seen with a finite level of detail, has no pre-determined coordinate system, and so on.

Another problem with concepts concerns the abstract ones. While one might reasonably suppose that many concrete concepts ("table," "sky," "tree") could be formalized through their connection to sensory experience, what about "university," "appointment," or "tendency"? As far as I know there is very little in the literature regarding how to formalize such concepts. In fact, as cognitive linguists rightly point out, understanding such words often implies understanding a whole body of knowledge (what they call a "frame")—an even bigger formalization problem.[9]

5.3.2 Having to

The problems of semantics go well beyond individual words or concepts and extend to fundamental dimensions of language like modality. For example, consider what "have to" means. To try to get a handle on this, let's ask a simple question:

> If I borrow a book from the library, do I have to return it?

Well, on one level the answer is yes, of course. But on another, one does not really *have* to return it, it's just that there may be certain negative consequences for not doing so. But are there not negative consequences for not doing a lot of things, many of which we would not say we *have* to do? Not watching the basketball game tonight may have the negative consequence that I don't get to enjoy the game, but clearly I don't have to watch the game. Another tempting idea is to say that "having to" involves some sort of social obligation, but at best that's only part of the answer; it seems reasonable to say "hang on a minute, I have to close my windows" when your friend knocks on your door and invites you to go somewhere, but rain is expected. Closing your windows can't be seen as a social obligation.[10] So what's *really* going on here?

I think we can view it like this: there are some things we are committed to achieving or avoiding—perhaps that commitment is social in nature (a commitment to our fellow man), or perhaps it is just something that we have attached a certain level of priority to, such as not letting the elements into our house. But not all commitments are the same—obviously one cares more about some than others, consequences for breaking them vary, often a commitment can be discharged through negotiation rather than by doing it (i.e. asking for an extension), and what one felt committed to in one reasoning context, or on one day, can change with circumstances. (If one has to get to the hospital, the windows can wait.) So the idea of "having to" do something is actually *relative* to some set of commitments that we have adopted in general, or one that society has adopted by convention, or one that we have adopted for the purpose of the reasoning episode we are currently engaged in. This view of obligatory modality is starting to look substantially more complex than any formalization I'm aware of.

Incidentally, what's the point of adopting a set of commitments and using a "have to" predicate, rather than expressing our thoughts in a more "fundamental" form like "If I do not return the library book then I may receive a bill for its cost"? The former form simply makes it easier to reason (and communicate with others). One has fewer contingencies to keep track of. Of course, when one decides one is unhappy with what is achievable within the "frame" of commitments one is using, one is always free to look for ways to drop commitments with acceptable costs. (This is an important aspect of human thought: if you can't solve the problem the way you've framed it, see if you can re-frame it.)

In light of this analysis, we can see that any simple analysis of obligation in first-order logic (perhaps in terms of a predicate $HaveTo$ that could be used to write "Sean has to return his library book" as (2)) is

[9] For an overview of frame semantics, see Croft and Cruse (2004).

[10] I guess it could if you rent the house, or if your spouse insists that the windows be closed, but these technicalities are beside the point.

out the window. Such a simple formalization does not embody the idea that the notion of "having to" is relative to an implicit background of commitments that can be investigated and modified.

$$HaveTo(Sean, Return(LibraryBook1)) \tag{2}$$

5.3.3 Semantics and AI

The point of the above discussion is that how to formalize semantics is a major, largely open, question that HLAI must somehow confront. Throughout semantics one finds nuances that don't have established formal descriptions, at least not with the level of operational detail an HLAI would require. There is a dearth of models both in AI (which, as I've already noted, has not for many years concerned itself with human or human-like thought) and in linguistics (where I suspect the obstacle is that linguists tend not to venture out into the rest of cognitive architecture, which is the part that has to actually *use* whatever representations they invent).

5.4 What do we need to know about thought?

Now that I've pummeled you with a laundry list of ways in which AI can't (yet) match human thinking capabilities, let me try to introduce some order by describing some of the main questions that an HLAI would need answers to and how we might go about developing them. Below are some of the main questions. Note that any of these questions could be asked both of humans and of hypothetical AIs, and the answers needn't be exactly the same. A HLAI needn't function exactly like a human mind if it can achieve the same effects.

- **The syntax of mentalese.** What is the syntax of mentalese—that is, of thinkable thoughts? The "syntax of thought" must of course be related to the syntax of natural language, but they are not the same, as any linguist will tell you. Actually, the issue is broader: the syntax of mentalese might tell us what the syntax of any "individual thought" is, but of course thoughts are organized together into larger structures—discourses, arguments, narratives, and so on. Individual thoughts can refer back to entities introduced earlier in a discourse, and can coexist within hypotheticals and other forms of shared context (cf. mental spaces (Fauconnier, 1994), discourse representation theory (Kamp, 1981)). So we'd like to know how the mind represents all these larger structures as well.

- **Inference and the control thereof.** What rules allow one thought to be inferred (or perhaps a better word would be "produced") from previous thoughts, or from sensory observations? If multiple inferences are possible (as they pretty much always are), what makes one better than another? The rules sought here are similar to the notion of inference rules in deductive systems, except that we don't want to think of inferences as coming with any guarantee of correctness (we should be able to go from "my car won't start" to "maybe the battery's dead because I left the radio on like I did last time"), or even to introduce any new propositional content at all (e.g. from "my car won't start" to "funny how that happened before in this same garage").

  There are two aspects to inference: some types of inference (like that second one) are "content-independent." Others involve the application of common sense knowledge, like the fact that leaving a car radio on could drain the battery. So we need to understand both the more "general" inference rules as well as how to represent, learn, and apply common sense facts.

- **Concepts.** Where do the primitive concepts (e.g. "water", "inventory", "borrow", "ready") come from? What is their structure? How do they connect to sensorimotor systems?

- **Entity tracking.** Obviously thought involves keeping track of various entities and what we know about them, and this includes both actual physical entities that we can see and touch as well as entities that we have only heard about or imagined. What sort of record is appropriate for performing these functions? At minimum, an entity record should contain the information necessary to build a referring expression[11] for the entity and answer questions about the entity by consulting memory and/or perception.

[11] For a rundown on various ways of referring, see for example Abbott (2010).

- **Where thought meets cognitive architecture.** How is the "train of thought" implemented in terms of cognitive architecture? How exactly does it meet perception and action? And how does it interact with memory? Which thoughts are remembered for how long, and how do they trigger replay of long-term memory? How do we notice parallels between different situations?

5.5 What's our approach to these problems?

We have now seen that there are numerous problems concerning semantics, reasoning, and thought for which we must have some strategy. This will be discussed more in the next section, but as before, to first order my answer to these questions is "look at what humans do." We will see examples of that proposal in action in the second part of this report.

Actually, to some extent there is already a field of study investigating what human thought looks like, namely linguistics (more specifically, semantics and pragmatics). So what should we do that linguists aren't already doing? I think we need to consider all of the above issues in the context of specific tasks that our hypothetical HLAI would actually perform. Whatever ideas we come up with for capturing the richness of thought, they need to be applied in the context of a behaving system to see if they actually work. It is this interface between thought and productive behavior that linguists don't tend to cross, but would-be HLAI researchers must. Additionally, we eventually need to think about the automated acquisition of mentalese and "rules of thought" from experience, since if the complexity of natural language grammar is any guide, mentalese is probably too complicated to model by hand.

6 The road to HLAI

So far we have discussed some of the major missing pieces for HLAI, two of the most prominent being rich thought processes and acquisition of conceptual, symbolic knowledge through experience. With those in mind, we can now step back and consider what sort of strategies might make sense for attacking HLAI. If we were to actually take HLAI as a serious goal, what strategy could we plausibly use to attack it?

6.1 Preliminaries

6.1.1 Baby machines

The first thing to be said about building a human-level AI is that we must build a baby machine—that is, a system which initially knows nothing (or very little) about the world, but gradually learns through experience. This approach to AI was first proposed by Turing (1950), and it is basically unavoidable. Humans possess a vast amount of what is often called "commonsense knowledge", which includes not only facts like "unsupported objects fall" but also an understanding of what words like "unsupported", "object", and "fall" *mean*. This knowledge is not only vast, but it also exists in a form to which we have no conscious access (quick, try to define "object" without using any of its synonyms). Typing all commonsense knowledge into a computer by hand is, for all practical purposes, impossible, and so any generally intelligent system would have to learn it *after* being turned on, just like human babies do. Many other researchers have come to the same conclusion; see for instance Guerin (2011).

6.1.2 The human blueprint

I believe that the only plausible way to build an HLAI is to model it, at least loosely, on the human mind. This is not to say we should try to copy every detail of the human mind (which would be impossible given our spotty knowledge thereof), but merely that the human mind should be the *starting point* for our designs. Anything we have a reason to change, we can and should change, but we should not try to toss the whole thing out and start over.

Here are some examples of the sort of architectural features that humans minds have that will probably need to be present (in some form) in an HLAI as well:

- A human mind has a train of thought which consists (as far as I can tell) of individual thoughts that are combinatorial structures not unlike sentences.

- Human minds have working memory and long-term memory, procedural and declarative knowledge.
- Human long term memory is associative and "content-addressable."
- Humans have a visuospatial sketchpad.
- Humans organize our moment-to-moment behavior hierarchically based on goals and subgoals. (E.g. if you want to check the weather, you create a subgoal of moving to the computer, which may have a subgoal of getting off your chair. Once you're at the computer you create the subgoal of opening a web browser. And so on.)

The reasons for basing HLAI on humans are simple. First, humans are the only existing example of human-level intelligence. Second, no one has proposed a plausible strategy for constructing a human-level AI *other* than drawing inspiration from the human mind.[12] For these reasons, whenever we are faced with a question about HLAI, we will ask "what do humans do?" to get a starting point, which we will then modify as appropriate.

We'll start using this principle right away by asking what overall architecture could work for an HLAI. That prompts us to ask what the overall architecture of the human mind is, which we address in the next section.

6.1.3 Core cognition vs. sensorimotor processing

There are various ways of analyzing the mind into component pieces, but one of the most important is the division between what I call "core cognition" and "sensorimotor processing."[13] By "core cognition" I mean the mental activities we would describe as "thought" or "reasoning." Such activities are deliberate, can be done without interacting with the external environment (e.g. lying in bed with your eye closed), and involve the generation of new ideas, thoughts, or other mental content. These activities are furthermore characterized by serial processing (often described as a "train of thought") that is introspectable to some extent (we can usually report the ideas our mind came up with, if not the exact combination of factors that caused a particular idea to arise). I would also count the deliberate initiation of motor behavior as part of core cognition, but not the subconscious processing that maps the high-level intention to muscle motions.

The other part of the mind is "sensorimotor processing," by which I mean the computations that take place in our brains, subconsciously and automatically, when we perceive the world or interact with it by moving our muscles. The computations involved in sensorimotor processing are no doubt extremely complex (as evidenced by our inability (thus far) to replicate them in computers, as well as the large proportion of the brains devoted to them), but at the same time they are fast, parallel, and effortless. Unfortunately we have no introspective access to these computations whatsoever (that is, no ability to describe the computations our brains use to translate incoming light into symbols like "chair" or "round" or to translate our intentions into the nerve impulses that flow to our muscles).

The division of the mind into core cognition and sensorimotor processing is not black and white; obviously the two systems interact and at the boundary we find phenomena that involve aspects of both systems. For example, the mechanism by which our brains process and generate natural language is not available to introspection, but that mechanism is at least somewhat involved in core cognition (which we can see from the phenomenon of inner speech). Similarly, when we deliberately visualize something (real or imagined) or imagine an action, we are engaging core cognition, but presumably core cognition is reading a signal that (in ordinary action or perception) would be coming from the sensorimotor component of the mind. But despite these gray areas, I think the distinction usefully carves up much of the computation done by the mind into one type or the other.

[12] AIXI (Hutter, 2001) and genetic algorithms are sometimes brought up as potential paths to HLAI; these ideas are interesting on their own merits, but neither is plausible as a path to HLAI. The problem with both is that they assume the availability of effectively infinite computational resources. Both ideas, in some sense, say "try every possible computational system and reject the ones that aren't intelligent." In other words, guess at random and pray!

[13] I made up those particular terms, but the idea is far from original; see for example Jerry Fodor's "modularity of mind" theory (Fodor, 1983).

6.1.4 Sensorimotor Prostheses

Given our view of the human mind as the logical template for HLAI, our initial assumption might be that an HLAI too should have distinct systems for core cognition and sensorimotor processing. However, I think we can improve on this design by essentially *eliminating* the sensorimotor processing component of the mind—that is, by building an HLAI with little or no innate capacity for vision (or other sensory modalities), and little or no innate capacity for motor control.

This idea may seem preposterous, but bear with me. First note that I am *not* proposing to build a system that is unable to interact with the external physical world. The ability to engage in sensorimotor interaction with the world (or at least a simulated world) is crucial both for performing useful tasks and for learning. The proposal is merely to alter the *mechanism* by which the AI accesses the environment. Instead of having automatic, "direct" access to the physical world the way that people do, our hypothetical AI would instead interact with the world through a computer (which would in turn connect to a robot). So if the AI wanted to gain information about what the world looks like, it would instruct the computer to tell it the color of a particular pixel seen through the camera. If it wanted to perform a physical action, it would tell the computer to move a particular actuator to a particular position.

Now, obviously it would be terribly inefficient for the HLAI to perform all its perception and manipulation on a pixel-by-pixel basis; it would be breathing through a straw. Fortunately, in the long term, it needn't do this: it could simply program the computer to do increasingly sophisticated processing. For example, with clever programming, it could outsource the identification of objects, or the grasping thereof, to the computer, so that instead of asking for the color of individual pixels it could ask for an inventory of objects in the image. The HLAI would read individual pixels only as a way of bootstrapping to the point where it could successfully write computer programs to do this low-level work on its behalf.

In other words, the proposal is that over time the HLAI would build up a "sensorimotor prosthesis"—a computer program that would play the role that the sensorimotor processing component plays in the human mind. This plan would eliminate the need for us to construct a human-level sensorimotor system ourselves (a problem we are nowhere near solving), which would save a lot of work and therefore make HLAI much more tractable.

Going one step further, we should ask ourselves whether natural language processing (and natural language generation) isn't susceptible to the same sort of simplification. I believe that it may very well be. Instead of trying to build natural language systems to perform the initial stages of language processing (such as parsing, disambiguation, and referent resolution), why not invent an artificial, constructed language[14] by which we would communicate with our HLAI? We could construct this language, which I call a "user interaction language" (UIL) in such a way as to avoid the aforementioned issues. Over time, the HLAI could figure out for itself how to map natural language into its UIL, through a process not unlike what a human linguist would do.

Interestingly, the "sensorimotor prosthesis" strategy outlined above is almost opposite to the conventional wisdom among AI researchers, many of whom seem to believe that to build HLAI we will first need to get better at things like vision, motor control, and natural language processing. I believe we should de-emphasize all those things and instead focus our attention on the problem of core cognition. We will now take a closer look at core cognition.

6.2 Core cognition

To recap the argument so far, we have proposed that the human mind should be our jumping-off point for designing HLAI, that the mind can be profitably divided into core cognition (i.e. thought) and sensorimotor processing, and that the latter may not be needed in an HLAI. As for core cognition, we must find a way to build a system that functions like human core cognition (likely with some modifications). In this section and the next, we try to characterize the basic features of human cognition, so that we will know what space of possibilities we are working within and also what major questions need to be answered.

What *is* core cognition? How does it work? For that matter, what would constitute a reasonable answer to those questions? Cognition is a computational process, so we can look to the way we describe *any*

[14]There is actually a community of hobbyists who build constructed languages or "conlangs," of which Esperanto is the best known example. The sort of constructed language we have in mind is similar in spirit, but would be different in various respects to be discussed later.

computational (or for that matter physical) process: we specify a state space and dynamics. In other words, we specify what states the system can be in at any given moment, and we describe how the state of the system changes from one moment to the next. For example, the state space of a computer chip is (roughly speaking) all the possible combinations of values that could be stored in its registers, and the dynamics are the rules for what the register values will be at one clock cycle based on what they were in the previous clock cycle.

In attempting to describe the state space and dynamics of a system, we must decide what level of abstraction to use. That is, how much should we "idealize" things for the sake of simplicity? In physics we can choose to conceptualize of the state of a rigid body as the positions and velocities of all its constituent atoms, or we can choose to assign just a few numbers for the position and velocity of the body as a whole.

When it comes to the human mind, we can think of its state as the state of all ∼100 billion neurons in the brain[15], or we can work at a higher level of description. At present, the latter approach is the only promising option, because we don't know nearly enough about the brain to build an accurate model of its behavior at the neural level. So we must look for a higher-level description of the mind.

6.2.1 Physical symbol systems

Fortunately, the mind (core cognition anyway) *does* appear amenable to description at a higher level: it appears to be what Newell and Simon (1976) called a "physical symbol system." Roughly speaking, this means that the mind operates by manipulating relatively small, combinatorial structures that look vaguely like language. To give a grossly oversimplified example, the idea is that when a person sees (say) a cat sitting on a table, his mind creates a data structure that looks something like this:

```
On(Cat, Table)
```

`On`, `Cat`, and `Table` are symbols, and `On(X,Y)` indicates that `X` is on `Y`.

Action is produced by similar-looking structures. For example, the behavior of moving towards something you want can be produced by something called (appropriately enough) a production rule:

```
If Want(X) and Location(X,Y), then MoveTowards(Y).
```

This says that if you want something, and it's at a certain place, move to that place.

Obviously there is much more nuance to the story, especially regarding these structures map to the real world. (What exactly does `On` mean? How was the cat recognized as a cat in the first place? Once one decides to move towards something how does one do it?) But they do convey the gist of how a physical symbol system functions.

In case the above explanation has left you wondering what would *not* be a physical symbol system, a common example is a neural network trained for some sort of perceptual or motor task. Neural networks represent long-term information through large sets of numerical parameters that describe the strengths of connections between neurons, and they represent short-term information through another large set of numbers describing the activation level of each neuron at any given time. Neural networks have proven their usefulness for some sensorimotor tasks (the things we've relegated to the sensorimotor prosthesis), but it is not at all clear how to build (and then train) a neural network that could reason.[16]

There are several reasons for thinking the mind can be successfully modeled as a physical symbol system, beyond just the lack of good alternatives. First of all, there are existing systems—known as "cognitive architectures"—which have gone a long way towards describing the mind computationally. In my view, the most veridical of these systems is ACT-R, which has been developed by psychologist John Anderson and collaborators since the 1980s. Anderson (2007) presents arguments for this approach, and ACT-R has been used to build plausible models of human behavior on numerous tasks, for example arithmetic and solving equations. Second, symbol manipulation is the basis for several things which resemble reasoning: computer programs, linguistic theories, and mathematical proofs. Third, it has been my experience that symbol-manipulating systems are workable models for any cognitive task one cares to study, and in the rest of this report you will hopefully come around to the same belief!

[15]Of course we'd also need to make a decision about what level of abstraction to use for the state of each neuron!

[16]Unless one starts off with a symbolic system and constructs a neural network equivalent, which doesn't buy us anything.

So let us adopt the "physical symbol system hypothesis"[17] for now and assume that the mind is best understood as a symbol-manipulating system. By the way, we needn't assume that the system is *all* symbols—some numerical parameters are necessary as well. This is essential to account for things like the fact that more frequently accessed pieces of information become more likely to be retained, and more successful actions become more likely to be tried in the future. The "successfulness" of a production and the "retrievability" of a memory are most logically modeled as numbers. Our knowledge must also have a probabilistic component to capture the nondeterministic aspects of the world. These so-called "subsymbolic" aspects of cognition are found in models like ACT-R too.

6.2.2 Symbols and GOFAI

Since we have placed an emphasis on symbolic computation (with some subsymbolic elements), some readers might be wondering whether we are just proposing to resurrect "good old-fashioned AI" (Haugeland, 1989). Yes and no. Certain aspects of GOFAI were problematic, certainly, but in many respects, GOFAI was much closer to a sensible attack on HLAI than modern AI. There is no way around the fact that thought is fundamentally symbolic; AI has shifted away from symbolic reasoning only by choosing to mostly attack problems that don't require human-like thought. That being said, many elements of the strategy we suggest differ from the GOFAI way of doing things. Here are some examples:

- We want to model the mind, not the world. So for example, we are much more interested in the process by which a mind would figure out how the physical world behaves than in trying to formalize naive physics ourselves, which was a typical GOFAI ambition (Hayes, 1990).
- GOFAI placed insufficient emphasis on learning (and the related problem of symbol grounding); we want to understand how the representations relevant to every aspect of cognition can be acquired automatically.
- Many GOFAI-era representations and formalisms did not reflect the richness and complexity of human thought but were more like crude approximations to it; we view this as a key area to be improved upon.
- A key aspect of GOFAI was system-building, and in general the resulting systems were not particularly mind-like (this was true of SHRDLU (Winograd, 1972) in many respects, for example). Our methodology (explained in the next section) is more focused on the analysis of human-like thought than on building systems, especially if we are not yet ready to make them cognitively plausible (see §7.4).

6.3 The state space of the mind

We have decided that our model of core cognition will largely involve the manipulation of symbols; we must now return to the question of its state space and dynamics, since these are what we want to replicate. Quite a bit is known about this question from cognitive science. However, we are still far from a complete understanding. I will begin by talking about what we do know with some confidence about the state space and dynamics of the mind. Then I will talk about what we don't know, as well as the very likely possibility of things we don't know we don't know. After that, in a sense, the rest of this report will be about ways to try to fill in these blind spots.

Let's begin with what science has to say about the state space of the mind. This is basically the question of what types of memory people have and how each type works. There is no doubt that there are at least two memory systems: working memory and long-term memory. Working memory stores information while we are actively using it, and decays within a few seconds if not deliberately refreshed. Long-term memory stores information over longer periods of time (from seconds to a lifetime), but information stored in long-term memory must be "retrieved" (that is, swapped into working memory) before it can be used. Since the concepts of working and long-term memory are so essential to any model of the mind, we will now discuss them in more depth.

[17]So named by Newell and Simon (1976).

6.3.1 Working memory

The most influential model of working memory is that of Baddeley and Hitch (1974), later expanded in Baddeley (2000). This model postulates three cognitive systems that each supply a different form of working memory: the visuospatial sketchpad, the phonological loop, and the episodic buffer. You are no doubt familiar with the first two of these: if I ask you to visualize an "E", remove the bottom horizontal line, and say what letter results, the visualization you perform amounts to "drawing on" and then "erasing from" the visuospatial sketchpad. The phonological loop is what we use when we repeat a series of digits (e.g. a telephone number) to prevent ourselves from forgetting them.

As for the episodic buffer, it is a somewhat more nebulous idea, but I agree that there is a need for (at least) a third form of working memory. We need a way to store propositional content so that reasoning can proceed using that content. Basically, in my view, this third form of working memory is where we store a small number of propositions that we have recently learned or retrieved, and when we generate a new proposition from one or more old ones, the old ones must be in this buffer. By way of example, if we have in this buffer the facts that "Mary booked a flight to Paris" and "Mary left for her trip last night" we can generate the conclusion that "Mary is probably in Paris now."

There is also, of course, a need to remember what you're doing at any given moment—that is, what one's current goal is and what broader plan or goal that goal is part of (e.g. "I am walking to the cupboard so I can get a drinking glass"). What exactly to call this type of memory and how much it has in common with other forms of working memory is unclear; Baddeley's three forms of working memory don't seem to encompass it, though he does postulate a "central executive" coordinating behavior which might be interpreted as stateful. Anderson (2007) includes a "control module" holding this sort of information. Actually there may be more than one form of this "goal memory." On one hand one has the intricate sequence of subtasks that any complex behavior (such as arithmetic or tying one's shoes) involves; on the other is one's tendency to keep doing an activity one has begun—for example if one has begun making dinner or working on a manuscript, this activity seems self-sustaining for a significant period of time. It does not have the decay characteristics of other forms of working memory (in fact it can be hard to "change gears" from one activity to another).

6.3.2 Long-term memory

The basic facts of long-term memory, familiar from both everyday experience and cognitive science, are that things we consciously perceive[18], as well as mental content we ourselves generate (such as mental images and thoughts) are often retrievable sometime later. Phenomenally, this retrieval resembles re-experiencing the original stimulus, and it is often triggered in an associative manner, meaning that a memory is retrieved after something related to it comes up.[19] Long-term memories cannot influence cognition without being retrieved, but once they are retrieved they sit in working memory at least briefly, and during that time they can do so. Moreover, retrieval of a memory strengthens the memory and makes future recall more likely.

As for the *form* of content in LTM, I think the "chunk" model used by ACT-R is reasonably accurate. In this model, memory takes the form of records not unlike database records or structures in a programming language. A chunk has a type, a number of fields, and values for those fields. For example one might have a chunk of the type "AdditionFact" with fields "addend1", "addend2", and "sum", the values being 3, 4, and 7 respectively. Chunks can also refer to other chunks to form more complex structures. (Note that the symbols that occur in chunks needn't be words, even though we tend to write them that way. One could just as easily have a symbol that stands for a certain type of texture, and when accessed causes one to visualize that texture—there needn't be any word associated with the texture.) Now, there is surely more to chunks in the human mind than this simple model accounts for, but the basic scaffold seems like a good start.

The fact that information goes back and forth between working memory and LTM would suggest that the two must have some relationship. I tend to think of LTM as a large pile of chunks onto which *every*

[18]What does it mean to consciously perceive something? I've concluded that what people usually mean by this idea (and for that matter many uses of the word "conscious" and "attention") is two things (in terms of observable consequences, not brain mechanisms): the thing being perceived becomes capable of triggering reactions beyond those that are "just automatic," and the perception enters long-term memory (though how long it sticks around can vary). The former of these effects, incidentally, I think is the essence of Bernard Baars' Global Workspace Theory (Baars, 1993).

[19]Even when we deliberately retrieve something, the retrieval often follows a related experience—for example, if I ask you to think of blue things, you may visualize the color blue before any particular blue thing comes to mind.

chunk that appears in WM (or enters core cognition directly through perceptual channels) gets dumped. Most of those chunks don't last long (perhaps only a few seconds); those that do are the ones we describe as memories.

One other aspect of LTM is its role in continually checking perceptual input against expectations. For example, if we walk into our home and something is out of the ordinary, we are likely to notice. The logical mechanism for this would be for LTM to continually play back recent memories from the same place "in sync" with actual experience (as opposed to playing back experiences from some other time and place) so that discrepancies could be noted.

6.3.3 Procedural memory

In addition to conscious, describable memories, we also have memories of *how* to do things (also known as procedural knowledge). In ACT-R, procedural knowledge is modeled by so-called "productions": rules which indicate what action to take in what circumstance. Typical actions are launching a new subgoal or querying memory. When we develop cognitive models in later sections, we will make similar assumptions.

6.3.4 Other forms of memory?

Besides working and long-term memory as just discussed, it seems to me there must be some other forms of memory in the brain. Some of these forms of memory exist purely within the sensorimotor systems (for example iconic memory (a form of visual memory with subsecond decay) and whatever "buffer" stores motor plans that are about to be executed), and these needn't interest us too much here. But there are some others that may be of more importance to HLAI. One is the memory that our "language module" presumably uses during the processing of incoming sentences. To parse (or generate) a sentence, it is necessary to retain a representation of one's current position within the sentence structure; given that we have little or no conscious awareness of the operation of our internal parsers, it is possible that this type of information requires some form of memory distinct from WM.

Another type of long-lived state that seems like it could plausibly exist in the mind is a sense of "orientation." At a basic level this would mean where one is (e.g. "at home" and "in my easy chair facing the fireplace"), but it could also be construed to include factors like whether one is working, or how much time one has before one's next appointment, or where one stands in some ongoing "script" that one is participating in (Schank and Abelson, 1977).

What reason is there to call this a distinct form of memory? For one, it's extremely reliable. Consider the following thought experiment. Let's say you are lost in thought, oblivious to your surroundings, and have been for some time, meaning that any chunks related to your surroundings that entered your LTM would have had to have done so some time ago. Suddenly the lights go out and you can no longer perceive your surroundings. Is it possible that you might then not know where you were (e.g. what room of your home)? This seems basically impossible for a healthy adult. But if our only memory source is LTM, then the chunks storing updates to our current location must be treated quite favorably by LTM, so as to be reliably retrieved when other types of chunks are not, especially since we don't deliberately try to notice or remember what room we've just entered.

Another reason to potentially treat orientation as a distinct form of memory is that there's a very distinct sensation of confusion when you are not in the orientation you thought you were (this might happen e.g. when waking up in an unfamiliar environment).

You may have also experienced the phenomenon of remembering where you were when you read something, or worked on a particular piece of writing; the more extreme version of this is remembering where you were when the Twin Towers fell. In such cases it seems that information has been firmly imprinted in memory without any deliberate effort to pay attention to that information and without any causal relationship between the events recalled and the place you were when they took place.

Actually, it may be more appropriate to think of this "orientational" memory as a sort of "extreme" form of LTM, one that is *always* recording (while awake and not visualizing some other location, anyway) and therefore extremely reliable. This idea also gels with the tight association found by neuroscientists between the brain's hippocampal structures (responsible for formation of long-term memories) and navigation.

6.4 What we don't know

We have now discussed in broad terms the structure of the human mind (core cognition anyway) and how researcher have modeled this structure in existing cognitive architectures. We will be adopting all of these architectural features and modeling choices, to a greater or lesser extent. But obviously existing cognitive architectures, and our existing understanding of the mind, aren't enough to build HLAI. So what's missing from that understanding? I think there are a couple things we know we don't know, and probably quite a few we don't know we don't know. This section will discuss these unknowns, and later sections of this report will speak to how we might start to figure them out.

There are two major places where we have known unknowns. One set of "known unknowns" is the mechanisms brains have but cognitive architectures don't, at least not in a sufficiently sophisticated form. A second known unknown is that cognitive architectures aren't a complete answer to how thought works but more of a toolkit out of which answers can be built; work must be done to bridge the two levels of abstraction. I suspect that that work will necessitate changes to the cognitive architecture, and that is where we should expect to find "unknown unknowns."

6.4.1 Known unknowns regarding learning

There are a few areas in which current cognitive architectures seem to have gaps. I will pick on ACT-R here (only because I like it so much!). One major issue is learning: to develop a cognitive model in ACT-R, the programmer must specify what chunk types are allowed, what fields they have, what productions are to be used, and so forth. In a human, of course, all of this data is automatically acquired, and the same will surely be true for HLAI. While ACT-R has made some proposals regarding learning over the years, they are not entirely convincing and in any case only cover small parts of the problem. This is not to fault ACT-R; clearly this is a hard problem, and perhaps one we can only make headway on once we have a better handle on just what sorts of chunks and productions we actually need (see §6.4.2). A second gap in ACT-R concerns things like curiosity, intrinsic motivation, and noticing violations of expectation—not surprisingly, these are all tied to the learning issue.

6.4.2 Higher levels of abstraction and unknown unknowns

Another respect in which current cognitive architectures are insufficient for HLAI is that they sit at a relatively low level of abstraction. In other words, it is not so much that current cognitive architectures are "wrong" as that they only provide the elementary building blocks *out of which* something resembling thought must be constructed. Chunks and productions constitute a highly general formalism (not unlike logic) which can model all sorts of things, and it is up to the user of the cognitive architecture to design the chunks and productions to get any particular behavior.

As a result of this generality, much of the problem of HLAI is to make choices about the design of chunks and productions, in such a way as to get human-like thought and reasoning. Note that we do *not* propose to hand-design all the chunks and productions our system will need; in keeping with our baby machine philosophy, we will need to find a way for the agent to generate new chunk schemas, chunks, and productions for itself. But even so, we do need to develop quite a bit more understanding about how chunks and productions connect to thought, reasoning, and language. So that is one of the main problems a strategy for HLAI should address, and one of the chief aims of the methodology we discuss in the next section.

Actually, I suspect that by the time we are through figuring out how to make thought—especially the rich forms of thought that appear on complex tasks—emerge from chunks and productions, we may be forced to make some rather substantial additions to the cognitive architecture itself. But those changes are not at all obvious yet, and so this possibility is a potential source of the "unknown unknowns" mentioned earlier.

6.4.3 Some concrete questions: the formation of memories of experience

To illustrate the points made in the previous section, let me give some examples of questions we might ask about how the chunk formalism should actually apply in practice—in other words, how to map the low level "chunk" concept to the rich mental phenomena we observe in everyday life.

Consider the problem of representing an *experience* (e.g. your vacation to Hawaii). Clearly this representation must be made of numerous constituent parts, but how are they organized? Perhaps something like the below could be used for each constituent?

```
Event
  Part-of: p
  Before: q
  After: r
  Type: Drop
  What: glass
  Onto: floor
  Actor: Joe
```

But still we have many open questions. When and how is an atom (`p` in this case) that links all the various subparts together *born*? Are memories strictly indexed as being part of *one* macro-event, perhaps with multiple levels of hierarchy? Or could they be part of multiple macro-events?

And to what extent should our memories directly reflect sensory impressions as opposed to higher-level concepts we infer *from* those impressions? If I remember teaching a class today do I have a chunk of the form `Teach(me, class, today)`? Or do I merely remember the experience of teaching it? Or both? How and when does one's mind take an experience of some sort and summarize it in semantic terms?

And how is it we can remember an experience as a unified "whole"? After all, any experience is extended in time and has many subparts, and so any LTM record for the experience as a whole must be a sort of impressionistic summary, capturing things like where and when it took place and "how it went" in general terms. How and when is such a summary formed?

A related question is what determines the *boundaries* of experience that say two "quanta" of experience belong to the same "scene" within a larger "album"? It's not hard to guess at this in general terms; particular chunks are likely to be tied together by similarities in the overall context: time, place, people present, activities going on, and so on. But the trick is to formalize it.

As you can see, we can quickly generate a great deal of questions regarding how exactly to represent any given situation with chunks. HLAI will ultimately need answers to questions of this nature.

7 Methodology

In previous sections we discussed the big-picture strategy of HLAI: we need to build a baby machine, mainly focusing on core cognition, and its overall structure should probably be similar to that of the human mind. We have also seen that existing cognitive architectures only get us partway to HLAI; acquisition of symbolic knowledge is a major open question, and the problem of replicating the richness of human thought is not directly answered by cognitive architectures like ACT-R, basically because thought is a phenomenon belonging to a higher level of abstraction. So we have identified some of the principal work that needs to be done, but how do we actually go about doing it?

I think it is useful here to consider the process by which a baby machine AI would develop from its initial state to the point where it was competent to handle the difficult real-world tasks that humans do. I have caricatured this process in Figure 1. The structure of the baby machine allows it to learn, resulting in knowledge. Knowledge here can include both things we normally conceive of as knowledge (like how to subtract or what a computer is) as well as the various skills required for *thinking*. Knowledge then leads to behavior of all sorts, some of which is aimed at, or at least results in, additional learning.

The overarching goal of the HLAI enterprise is to construct a baby machine in such a manner that the development process just described is possible. What makes this such a difficult problem is that there are long causal chains from the structure of the baby machine through to the desired behavior that the mature machine produces (potentially stretching through many cycles of the learn/behave/learn loop), but it is only the end results that we can get a strong handle on. This is an important point, so I want to elaborate on it in some detail.

The key question as far as methodology is concerned is what tools we have at our disposal for investigating the bubbles of Figure 1. Neuroscience and psychology offer a whole arsenal of tools (everything from fMRI to

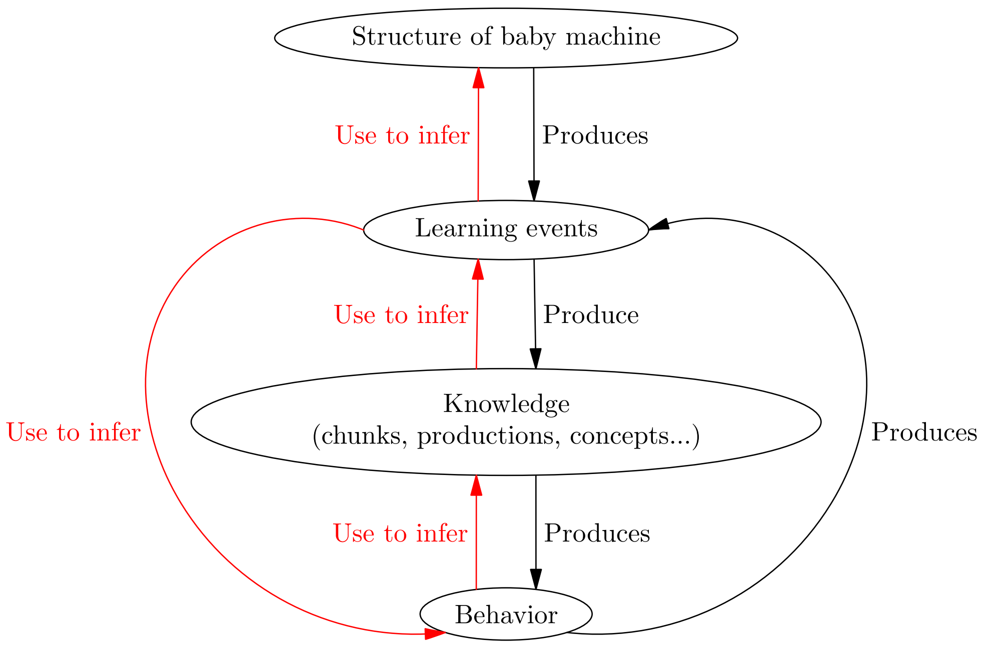

Figure 1: Causal chain by which a baby machine ultimately comes to produce intelligent behavior. Red links show a possible strategy for reverse engineering.

eye-tracking), but the disappointing fact is that *none of the fancy tools gives us what we really need, which is precise details about what our mental representations look like.* The only way to get that sort of information with any confidence is by either observing our own thought processes introspectively, or by investigating our usage of language. I call these modes of observation "DOT methods," for "direct observation of thought." DOT methods (as imperfect as they are!) are by far the clearest microscopes we have into the workings of the mind.

But even DOT methods have limitations. Of the four bubbles in Figure 1, it's primarily behavior that DOT tells us about. Knowledge is not too accessible to DOT methods because it is largely subconscious (declarative knowledge is not, but procedural knowledge, linguistic knowledge, common sense, and the *format* underlying declarative knowledge are). And the updates to knowledge that constitute "learning events" are likewise mostly subconscious. Certainly we do not have direct conscious access to information about the cognitive architecture we were born with. Moreover, DOT methods can't be applied at all to pre-verbal children (really it is questionable how useful they are when applied to young children in general[20]).

So the bottom line is that our only adequate tools of investigation, DOT methods, mainly tell us about the thinking of adult minds. We lack a direct way of investigating where that thinking came from, even though the origins are the important part. So to figure out the possible structure of a baby machine that could give rise to human-like thinking, we need to peel the onion and invert the causal chain that led to the thoughts we observe. We infer what knowledge led to the observed thought, where that knowledge came from, what behavior led to that learning, and so on. **That is our fundamental methodology: investigate human thinking, and work backwards to uncover knowledge and learning mechanisms that could plausibly produce it.** This process of investigation is indicated in Figure 1 by the red arrows. In the interest of having a name, let's call it TKL methodology (thought, knowledge, learning).

Note that the developmental pathway we construct to get to human thought needn't be the same as a human's, we just need *some* path back through time from human-like thought to a baby machine. Further-

[20]For one thing, by the time a person can insightfully observe and report their own thoughts, they have basically reached adult-level thinking skills. One could (as Piaget did) investigate the sorts of things children say and think before they have reached the rational thinking of an adult, but children's confused thinking—by the very nature of confused thinking—is difficult to model and difficult to "reverse-engineer" to figure out where it might have come from. It is also unclear that a HLAI *ought* to recapitulate anything quite like the stages of confused thinking that human children go through.

more, humans *learn* everything we know about how to think, but in designing an HLAI we might instead choose to build a baby machine with some degree of "rational thinking skill" built in. This approach might not come up against the primary obstacle encountered when one tries to formalize all of human knowledge, which is its sheer volume. Perhaps a baby machine could be a sort of "savant," ignorant of the world but skilled at figuring things out scientifically. I think the jury is still out on the right balance between learning to think and knowing how to think (and what parts of learning to think would happen when). This is the sort of thing that working backwards from human thinking should ultimately clear up.

7.1 Task analysis and storyboarding

We have already explained that we want to work backwards from observations of human thought to infer the underlying knowledge and the mechanisms by which that knowledge could have been acquired. But operationally how do we do that? Essentially we need to pick some sort of task or behavior we might want an HLAI to perform, perform that task or behavior ourselves, and make note of what thoughts (and behaviors) arise as we do it. Then we have some raw material with which to drive the reverse-engineering process. I call this process "task analysis."

The emphasis on *tasks* here is deliberate and important. Just haphazardly observing human thought in many unrelated contexts, while potentially useful, is not going to lead to a picture of how all the various components of cognition are *integrated* to perform useful work. For that, we need to see all the elements of thought working together and make sure our models of those pieces, together, enable the task to be performed.

Task analysis can proceed at various levels of granularity and formality. The ideal is to come up with a fully formal representation of our thoughts, but of course we must begin with informal observations, and any representation that we come up with will be somewhat speculative. (But we can make good guesses—as linguists routinely do!—about the formal structure of thoughts, and since we don't need to copy human thought exactly, some degree of mismatch with "true" human thinking is acceptable.)

Importantly, in constructing formal representations of episodes of human thinking, we will often need to fill in details on a "best effort" basis. It is impossible to tell *exactly* what one's mind is doing—at best, one can get a rough sense that allows various possibilities to be judged more or less true to life. But to proceed with reverse engineering particular episodes of thought (to get at underlying knowledge and learning processes), it is necessary to have firm ground to stand on: we need specific descriptions of the thoughts and behavior that took place. Therefore, we will have to take whatever we can glean from DOT methods and fill in the details as best we can to obtain an account of mental computations that results in the successful performance of whatever task or behavior we're investigating. I call this process "storyboarding," because we are in some sense storytelling: we want to construct a narrative that feels true to life and which gets where it needs to get. And as with storyboarding in real life, we will often want to fill in the details of a sequence of mental events only partially at first, increasing the level of detail over time.

Actually, even in the end we needn't aspire to model tasks *completely* (that is, model every substep), as long as we get the gist of the process. The point of storyboarding is that it gives us material to work from to theorize about acquisition mechanisms, not to actually build a working HLAI one piece of knowledge at a time. Some degree of judgment is required in deciding how thoroughly to model any (sub)task: we want to be complete enough to verify that our mechanisms are sufficient for the task, but not so complete that we are repeating ourselves and wasting time.

Don't be put off by the term "storyboarding." It does not mean we can just make stuff up. We are subject to two constraints: the constraint that the task must be accomplished by the mental events we postulate, and the constraint that those mental events should resemble what humans do. How to best fulfill the second constraint may be subject to some amount of disagreement between any two investigators—often we cannot be sure what our minds do, and sometimes it makes sense to do things a bit differently because a HLAI would have different strengths and weaknesses compared to a human.[21] But the fact remains that some stories are better than others. If we can write enough—and good enough—stories, we should eventually be able to work out the necessary knowledge representations and learning mechanisms for a baby machine.

[21]Humans have very modest working memory capacities, a limitation that would be artificial in an HLAI, but conversely, humans have much better sensorimotor systems than we are likely to be able to build anytime soon (see §6.1.4).

The methodology outlined above—task analysis (via DOT methods), storyboarding, and reverse-engineering—is what we will pursue for the remainder of this report. To get started on that, we'll need to think a bit about task selection: what are the most profitable tasks to analyze?

7.2 What task to work on?

Our methodology calls for us to "put the mind under the microscope"—we observe the mind working away and try to figure out what it's doing, or what it *would* be doing if it were a HLAI. For this exercise to be meaningful, we must pick some concrete subject matter for the mind under the microscope to chew on. What task should we have it working on?

There's no one right answer, of course. Different tasks exercise different aspects of cognition to a greater or lesser degree. But there are, I think, some logical rules of thumb.

1. Simple tasks are good. We don't want to start with something like "summarizing a newspaper article," which requires an extraordinary amount of background knowledge and linguistic ability. Small-scale puzzles and games which don't require natural language understanding, common sense, or sensorimotor skills are probably a better starting point.

2. The task should not be so simple that it can already be automated easily. For example, arithmetic is probably not a great choice: building a model of a mind that does arithmetic is relatively easy[22], and unlikely to address any of the serious obstacles to HLAI. We want tasks that are interesting enough to address "thinking gap" and "learning gap" alluded to earlier (§4).

One general class of tasks that meets these criteria is tasks requiring the agent to "figure stuff out" in a reasonably simple environment. Such tasks are promising on several counts: they can be constructed in a way that removes the complexities of language, sensorimotor interaction, and common sense; they allow for all sorts of interesting thought processes; they are decidedly nontrivial; and much of a baby machine's job will be to "figure stuff out," so we might as well get started on that.

"Figuring stuff out" can be operationalized in a variety of ways, and there have been many proposals along these lines in the literature. For example, we might want the HLAI to work out the rules of a simple game based on observation; Genesereth and Thielscher (2014) call this "inductive general game playing", and the GVGAI competition (Perez-Liebana et al., 2016) has a similar premise. Much of the work by Doug Hofstadter and his students also aims at figuring out underlying rules in various types of puzzles, including numerical sequences (Meredith, 1986; Mahabal, 2009), character string analogies (Hofstadter, Mitchell, and others, 1994), and Bongard problems (Foundalis, 2006). Markan (2016) proposes the task of figuring out Mealy machines with "natural" structure. Another domain in which we can study the figuring-out process is problems inspired by actual scientific goals, such as letter recognition. We will have much more to say about this latter idea starting in §10, and that will be the domain we pursue in subsequent sections. But keep in mind that any of the above tasks (and many others) would be suitable to our proposed methodology.

7.3 Storyboards as a middle road between programming and cognitive modeling

The TKL methodology—and storyboarding in particular—takes a "middle road" between two "extreme" ways of attempting to model thought, both of which were tried in the history of AI and both of which lost steam over time. We believe that a hybrid methodology addresses the problems each encountered.

The first way of modeling thought is the think-aloud protocols performed most famously by Simon and Newell (see e.g. Newell and Simon (1972)). Simon and Newell had subjects solve problems (generally some form of logic puzzle) while thinking out loud, recorded their thoughts, and constructed formal models of their thinking. If there is anything to be criticized in this work (which was extensive and careful) it might be that—at least for purposes of HLAI—Simon and Newell were too exhaustive in their analyses. They sought to figure out *all* the bits of knowledge that a given subject was deploying on a task, and to annotate every line of their transcripts with which bit of knowledge was used when. The consequences of this exhaustiveness

[22]It is not so easy if one wants to reproduce human psychometric data, but that is not particularly important if our goal is simply HLAI.

were that the methodology was extremely laborious, and only simple tasks (simple in the sense that any student of AI could automate them) were modeled.

Compared to the HPS approach, the TKL methodology takes a somewhat more relaxed view of what we need to figure out about humans. We don't need to analyze *every* step of an observed human problem-solving process, nor do we need to find *all* the knowledge involved. It is fine if our storyboards only cover *some* steps of human thinking, leading us to only infer *some* of the relevant knowledge. We don't need to get it all, we just need enough to enable productive theorizing about how to build a baby machine that acquires it. Moreover, as we have already mentioned, we are not so concerned with capturing *exactly* what humans do (certainly not what one single subject does), as long as our storyboards display human*like* thought. By making these compromises, the TKL methodology lets us explore a much wider range of thought more quickly, which is necessary if we are to address the critical "thinking gap" discussed in §4.

I mentioned there were two extreme ways of modeling thought; let's now consider the second. This extreme basically says that what humans do doesn't matter: just build *any* computational model that can perform the task you've decided to study. This approach mistakes the means for the end: studying particular tasks is a *means* of figuring out something about the mind that will apply *in general*—the individual tasks we study are not ends in themselves. This is actually the trap that AI as a field has been sucked into in the last few decades: whatever ambitions it initially had of figuring out what intelligence *is* were displaced over time by efforts to solve particular tasks, whether or not the methods used to solved them had anything to do with human intelligence in general. So as we pursue the TKL methodology we must avoid the trap of treating the solving of tasks as our primary objective.

7.4 On system building

Having explained what I think we *should* be doing, let me now discuss something I think we should *not* be doing but which is all too tempting: building systems (i.e. software programs that perform the tasks we study). Actually I need to qualify that statement, because I do think that *in principle* it would be highly desirable to build such systems, as the process of doing so would help confirm that we have done a good job modeling cognitive processes. The problem is that in practice, if one makes it one's goal to build a system to perform this or that task, typically one just ends up with a system that does the task in a manner that does not resemble the human mind at all. Such systems don't help us with HLAI. One might think they could—after all, there's no a priori reason why HLAI *must* be modeled after the human mind—but in practice, systems that don't resemble the mind never have the kind of generality HLAI demands (and more importantly, that's obvious at *design time*, it's not something that requires us to keeping building systems to find out). And it goes without saying that system building has a high opportunity cost.

So I think the correct strategy is this: if we can get to the point of building systems that truly model the "human way" of performing a substantive cognitive task, and it looks like building the system would help us validate and/or refine our (otherwise less concrete) cognitive models, then by all means we should do so. But we should not regard building a non-mindlike system, or one whose operation won't tell us much about the hard parts of cognition, as a substitute for more analytical work that does address those hard parts—even if the latter sort of work is preliminary and patchy at first. Of course, the work of studying human cognition may be regarded as *design* work for an actual, implementable system, and indeed a precise storyboard looks just like a particular *run* of a built system. The key distinction is that we are not in such a rush to build a system that we will accept just any system (or view the whole enterprise as a failure if we can't build a suitably mindlike one in the short term).

This philosophy of deferring system-building, while we instead study how the target system would hypothetically behave, may seem at odds with "normal" scientific methodology. But (as already noted) HLAI is such a stubborn problem that it needs an unusual approach, and besides there is certainly a precedent for the kind of analysis we propose. That precedent is linguistics: linguists study language (which is close to thought, albeit not quite the same) at an extremely fine-grained, formal level, but they are in no rush to build systems that can talk, or even to write down a complete formal grammar of any language (which is prohibitively complex). Not everything about linguistics directly maps to HLAI, but I think the spirit is similar.

8 Preface to the cognitive modeling sections

Now that we have outlined a philosophy and methodology regarding how to pursue HLAI, it's time to show how our proposals can be operationalized. The rest of this report will be a somewhat freewheeling investigation of the issues raised so far. We will be looking at a number of specific tasks we might give an HLAI and situations we might place it in, and the goal will be to work out—often by means of storyboards—something about how it might represent, reason, and learn. A significant chunk of the investigation will concern what I call "microscience": having the HLAI play the role of a scientist (more on that in §10). We will try to keep in mind the objectives discussed so far: being reasonably faithful to what humans do, exploring some of the "richness" inherent in thoughts and concepts, and paying attention to the acquisition of the structures we postulate. What we do should be regarded as a sort of "random sample" through the space of problems we could have worked on—in other words, I have not attempted to "survey" the space of possible projects that our methodology might suggest, only to point out a few of them.

Here are some of the major themes and questions we will see throughout this investigation.

1. Representation: What are good representations for things (concepts, situations, etc.)? We want representations that resemble those used by humans, insofar as we can determine how human representations look.

2. Ontology: What is the nature of concepts? How are they represented? How are they acquired?

3. Thinking: What sorts of thinking are required for performing "interesting" cognitive tasks (which is to say, tasks that haven't already been thoroughly explored and modeled, yet aren't hopelessly hard)? How are rules for thinking acquired and how can they be formalized?

We will cover two sorts of tasks. First is a relatively mundane task involving searching a list. We will use the "vanilla" version of this task to illustrate one of our modeling formalisms, and then a variant thereof to study representation, ontology, and planning. Second, starting in §10 we consider a task in which the agent is supposed to learn about data (in particular, images of letters), again discussing representations at length as well as low-level cognitive models and the nature of concepts.

9 Searching a list: a case study in thought and representation

In this section we will develop cognitive models for a simple task—namely, scanning a list for a target element. We will begin with a straightforward version of the task, using it to illustrate what a precise cognitive storyboard would look like, as well as to introduce our modeling formalism for such storyboards. We will then extend the task so that the agent is called upon to exploit prior information in executing the task. This will lead us to discuss representational issues at some length (so that we can represent the prior information in a principled and human-like manner). We will develop a framework for representing concepts and show how, using that framework as a foundation, we can assign meanings to propositions that resemble natural language. Having done this, we will discuss how the extended task might actually be executed, which will lead us to discuss the general nature of the connection between goals and plans.

9.1 The task

The task we'll consider is a simple one: we give the agent a list of integers and ask it to find a "1" in the list. To make this more concrete, we may think of the agent as interacting with its environment by exchanging messages. To convey the goal, we send it the message (3), in which `1` is the number to be found and `s` is a reference to the list in question. (You can think of our using `s` as a form of deixis.)

```
FindInList(1, s)                                                    (3)
```

Then the agent has the ability to issue commands like (4), which causes it to perceive what is at position 9 of the list.

```
Probe(s, 9)                                                         (4)
```

Note that we have assumed the agent can echo back a deictic reference it was previously given; this is a simple way to allow both us and it to interact with the same environment.

Our concern in this section is how to model the agent's thought and behavior on this task. For the "vanilla" version of the task (which is what has been described so far), the agent's job is simple and so we can easily construct a precise story about what's happening (at least at the level of abstraction and veridicality we care about). We will do that in sections 9.2 to 9.4. We do this not so much because the story is interesting but because we want to develop a formalism for writing such stories. After that, we will look at an extended version of the task where there are more interesting cognitive phenomena to explore. For that we will work at a less detailed level of description (though of course in the future it would be nice to cover the story more thoroughly and precisely).

9.2 Thought vs. running computer programs

How should we model the agent's behavior on the vanilla list-search task? Let's begin with how we would handle the task if we were just writing an ordinary computer program. Here's how that program might look (I have used Python but the choice of language is not important):

```
def FindInList(target, s):
    for pos in range(len(s)):
        if Probe(s, pos) == target:
            print pos                                   (5)
            return
    print "not found"
```

This is all well and good (and it illustrates essentially what we are going to have the agent do), but in general it won't do to regard the agent's procedural knowledge as merely a collection of programs written in traditional programming languages. There are a number of ways in which human thought processes, even when they are following some simple algorithm like the one above, go beyond what a computer would do in executing that same algorithm. For example:

- Humans will notice and course-correct if we get stuck in an infinite loop, if something is going slower than we anticipated, or if we find ourselves repeating some action which could go faster if optimized or done "in bulk."
- For that matter, as we work we may well notice and remember any number of things about the data we're processing that are irrelevant to the task at hand. For example, as we scan the list we might remember what numbers occur frequently. (I call this "extracurricular processing.")
- We can be interrupted and continue (perhaps leaving ourselves a note to remember where we left off).[23]
- We can talk about what we're doing: what our goal is, how far we've gotten so far, and why we're doing it the way we are. For example, in reading the list we know not merely that we're executing a "for" loop but that we are doing it *in service of traversing a list.*
- We can react to and exploit new information that comes in while we're working.
- We remember our actions and findings; if someone re-submits the same "find" command after we've done it once, we'll just report the same thing (or better, "you already asked that") without redoing the work.
- We can take our thought process as an object in its own right—something that can be examined, remarked upon, and critiqued.

Most of the above capabilities have no analog in any programming language I'm aware of.[24] Nor is it obvious how to extend the semantics of normal programming languages to get those sorts of behaviors. As

[23]Yes, computers do this, but not as flexibly.

[24]I don't mean here that one couldn't write a program with those capabilities (obviously I don't mean that, since that's precisely what an HLAI would be!), but rather that there is no programming language that allows you to write something like (5) and get all of the cited capabilities "for free," with no extra work.

```
Line  Label  Type      Parent   Details

  1    s     ENTITY             List
  2    g     GNSF        ^      Find(1, s); state = search
  3    i1    IT          ^g     pos = 0
  4    x1    GNSF        ^i1    Probe(s, 0)
  5    x1           ::          done 0
  6    i2    IT          ^g     pos = 1
  7    x2    GNSF        ^i2    Probe(s, 1)
  8    x2           ::          done 0
  9    ...
 10    x99   GNSF        ^i99   Probe(s, 98)
 11    x99          ::          done 1
 12    g            ::          result = 98; state = report back
 13    rb    GNSF        ^      Say(98)
 14    g            ::          done
 15    rb           ::          done
```

Figure 2: TLOG for a list-searching task.

discussed in §7.1, to figure out such extensions we need to develop storyboards in which an agent exhibits the desired behaviors, and then work our way to general principles. To implement that strategy, we'll need a formalism with which we can describe the execution of a behavior. And so to that end, we will now introduce such a formalism, using the example of the "find a 1 in this list" task. I call this type of representation a "thought log" (TLOG).

For now, we won't try to incorporate any elaborations into the cognitive process beyond what is already captured in (5). This means that our TLOG will be (roughly) a way to notate the states an interpreter moves through as it executes something like (5). Later we will consider some ways in which our TLOG formalism could be elaborated to reflect the sorts of considerations mentioned above. I should also note that our TLOG representations are not set in stone—we expect that settling on an optimal syntax and semantics will require much more investigation.

9.3 The TLOG formalism

To introduce the TLOG formalism we will work through a storyboard corresponding to a single run of our list-searching task. To do this, we need to specify what list is being searched and what the target is. We will assume the target is 1 and the list consists of 100 numbers, all of which are 0 except for a 1 in position 98.[25]

Our TLOG is basically a list of stack frames and other objects produced in the execution of (5). (I'll call stack frames "GNSFs" (goal node/stack frames), to reflect the fact that they typically correspond to goals to be achieved.) When a new object (GNSF or other) is produced, we add it to the end of the TLOG, and if an old object changes, we add an entry to indicate the change (we don't go back and change old entries). The full TLOG for our task is shown in Figure 2. Each line has a label identifying the object being created or updated and the data being associated with it. Objects list their types and parents (if applicable) when first created, and updates are marked with "`::`".

The computation starts with a GNSF `g` representing the overall job to be done, namely searching the list. `g` references a list entity `s`, so before our entry for `g` we have an entry for `s` (but how the agent came to be aware of it is not shown). When `g` is created, the caret indicates that this GNSF has no parent, and `state = search` indicates that we are in the initial stage of executing `g`, wherein we search for the target. (The second stage will be to report the answer back to the user.)

[25]We assume 0-based indexing.

To pursue `g`, the agent executes a loop. Iterations of the loop are treated much like GNSFs, except I label them as `IT`.[26] Our first iteration `i1` appears at line 3; the variable `pos` stores the list position being looked at, and `^g` indicates this iteration has `g` as its parent.

Within each iteration, the agent must execute a "probe" action to query what is at that position in the list. This occurs first as `x1` at line 4. We treat `Probe` as a primitive, so it immediately completes, leading to an update at line 5 where `x1` is marked done, with a result of 0.

Similar iterations occur starting at line 6, until we finally find the target at line 11 (note that line 9 elides most of the iterations). At that point, `g` updates to indicate that the answer is 98 and that the next thing to do is report back the answer. At line 13 `g` then triggers a `Say` GNSF responsible for reporting the answer 98. I have assumed tail call elimination,[27] which is why `rb` has no parent. For the same reason, as soon as `rb` is created, `g` may be marked done (which is why it is shown completing first at line 14). Since `Say` is a primitive, `rb` also immediately finishes at line 15, and the task is complete.

9.4 The "code" behind TLOGs

What a TLOG leaves implicit is the process by which each line is generated, which of course we also want to understand. There are a couple aspects to this question: one is *which* GNSF to work on at any given time, and another is *how* to work on it.

Tracking which GNSF to work on is analogous, in a typical computer program, to keeping track of what stack frame you're in (this is the role of the stack pointer). I have not bothered to include a stack pointer equivalent in this TLOG (or later ones), because it is generally obvious what to work on: just pick the most recent incomplete GNSF.

The second issue is the rules that govern how the agent pursues a given GNSF; these rules are analogous to productions in ACT-R, or to plain old computer code. In general, once we have a storyboard like that shown above, it is not too hard to invent rules that produce it. The "real" work, I think, lies more in coming up with a good storyboard. For that reason, I will not place too much emphasis on the rules behind TLOGs. Nonetheless, just to demonstrate the sort of rules one would use, here are some of the rules that would be involved for our TLOG in Figure 2:

Rule 1. If one is starting work on a GNSF of the form

```
g GNSF       Find(t, z); state = search
```

then create

```
  IT    ^g   pos = 0
```

Rule 2. If one is starting work on an iteration `i` in the context

```
g GNSF       Find(t, z)
i IT    ^g   pos = k
```

then create

```
  GNSF  ^i   Probe(z, k)
```

Rule 3. If one has just completed a GNSF `x` in the context

```
g GNSF       Find(t, z)
i IT    ^g   pos=k
x GNSF  ^i   Probe(z, k); done 0
```

[26] For various reasons, I have chosen not to handle iteration by just updating a counter variable stored in the GNSF, even though that would be a more faithful representation of what a computer does. One reason for this is that it seems to me that humans, at least some of the time, remember information about past iterations (especially if something notable happened during them). This suggests a representation in which separate iterations get separate records.

[27] See for example Abelson et al. (1998).

then create

```
        IT     ˆg    pos=k+1
```

Similar rules can be written to generate the entire TLOG.

The foregoing concludes our discussion of TLOGs for now. We will not make too much use of them in the remainder of this section, but we will come back to them in §11, when we look at thought processes involved in analyzing bitmap images. In the mean time, we will complicate our list-searching task and use it to explore questions about ontology, representation, and planning.

9.5 Making things interesting

Now that we have discussed in some detail the vanilla list search task, we will explore an extension, one in which the agent is already given some information about the list and should exploit that information to reduce the amount of work to be done.

Here is the new task. Suppose we tell our agent fact (6).

There are no 1s before position 50. (6)

If we then ask it to search for a 1, it should start at position 50, as any reasonable human would. How do we model the associated thought process? This is an interesting question because it gets at the issue of how to bridge reasoning and action. It shows that we can't use some simple program like (5) to represent what happens when we're asked to search a list, because we may need to do some thinking before we jump head-first into the "doing" of the task.

I think there are two main issues to be addressed. First is the question of how to represent (6). Second is how we infer from (6) a plan for executing the goal. We will tackle these questions in turn.

9.6 Representation

In keeping with our philosophy from §7.3, we want to represent (6) in a "human-like" manner. We cannot know *exactly* how humans represent such a proposition, nor can we *prove* that for the purposes of HLAI it is desirable to do things in a human-like way. Nonetheless, we can make some reasonable guesses about the human representation, in large part based on the way we verbalize things, and I think we can reasonably conjecture that making our representations human-like will ultimately pay off. In general, doing things "the human way" will allow for more compact representations and a higher level of abstraction, which we could expect would enable relevant patterns in such representations to be noticed more readily.

So what then is a "human-like" representation for (6)? We can develop one by starting with a not-very-human-like representation and improving it. Here is a standard way of notating (6) (using first-order logic with s-expression notation):

```
(not (exist i:INTEGER
        (and (< i 50)                                    (7)
             (= (element L i) 1))))
```

(Here `L` represents the list under discussion and `INTEGER` is a type annotation.) Note how distant this formulation is from (6). A human can easily read (6) and see what is meant, whereas (7) must be puzzled out. So let's see if we can find something more natural. We can start by treating "occurrences in the list of the number one" as *entities*. We'll use `(occurrence-of One)` to represent the type of such entities. Now we can write the proposition like so:

```
(not (exist occ:(occurrence-of One)
         (< (position-of occ) 50)))
```

(You may have noticed that the reference to `L` has been lost. Don't worry, it will come back in snippet (12).) Having done that, we might as well go one step further and view positions (like "position 50") and places (like "before position 50") as entities as well:

```
(not (exist occ:(occurrence-of One)
        (in occ (before (position 50)))))                    (8)
```

One final step: if we define a predicate as in (9), then we can strip out the variable from (8).

```
(no-exist-in A B) ::= (not (exist x:A (in x B)))             (9)
```

Now we have (10) as our final representation.

```
(no-exist-in
  (occurrence-of One)                                        (10)
  (before (position Fifty)))
```

This is quite close to (6); it simply makes the syntactic structure more explicit and therefore easier to work with. Now that we have it, we can also see why it may be preferable on practical grounds to something like (7). Because it uses "higher-level" notions (which we can see from its compactness), it is likely to allow higher-level, more specific inference rules which would allow reasoning to occur with fewer steps.

9.7 Semantics

So far we have not been careful about specifying the semantics of (10). Actually, for our present purposes we could get away with keeping things informal, since the reasoning we will pursue for the list task will work directly with (10). However, that's not entirely satisfying, and furthermore, addressing this issue presents a good opportunity to investigate how concepts might be formally represented (a question that was raised in §5). So let us try to supply (10) with semantics. We will begin by discussing the ontology that is relevant for this situation.

9.7.1 Ontology

Let's go through the ontology we'll be using as we talk about lists. I'll write types (or concepts, or whatever you want to call them) in all-caps. For each concept, we will write out what I call a "concept record," which specifies information that the agent understands about the concept in question. (I have not tried to be exhaustive in writing down this information, but I will hit the main points.)

Naturally, we'll start with `LIST`s. I will just assume for now that all lists are lists of integers. The first thing to note is that we are viewing lists as objects "out there in the world." That is, they are objects that the agent can probe (and potentially manipulate) through its interface with the world. This is a somewhat different conception of a list from the "mathematical" one, in which a list of integers is a mapping from $\{1, 2, \ldots, N\}$ to integers (for some N). In our conception, even if one knows all the elements of the list, one doesn't know everything about it, because one doesn't necessarily know things like what it's representing or where it came from or what the consequences of altering it would be.

Of course, ultimately we want our agent to understand *both* senses of "list," and how they relate (for example, a list in a computer program, insofar as it is a "storage device," stores precisely its elements and nothing else, so if we know the elements then we know everything there is to know about what it is storing). But these issues will have to be deferred for later; for now we will just say that a `LIST` is a "sensorimotor object"—the sort of thing the agent can recognize, perceive, and manipulate through an interface we will build for it. (We won't delve into the details of said interface, but note that it could be extremely simple, as suggested earlier at (4).) We will also assume the agent is aware that lists have a length and a collection of elements. The concept record then looks as follows:

```
LIST
  [sensorimotor object]
  length: INTEGER
  elements: collection consisting of all INT-OCCURRENCE[this]
```

This says that a list is a sensorimotor object, that it has a length (which is an integer), and that it has elements. The notation used for the `elements` field will become clear as we proceed.

Since we have referenced the `INTEGER` concept, let's also look at *its* concept record. Here we simply say that an integer is defined by its digits.[28] Furthermore, a `DIGIT` is a "sensorimotor symbol," meaning that recognizing and producing digits is handled in the sensorimotor interface.[29,30]

```
INTEGER
  digits: sequence of DIGITs
  uniquely defined by: digits

DIGIT
  [sensorimotor symbol]
```

When we write that `INTEGER` is "`uniquely defined by`" `digits`, we simply mean that there is at most one `INTEGER` with a given value for `digits`.

We can also speak of "positions" within a list. We will use a concept `LPOS` for this. A list position is in general defined relative to a particular list, a fact which we express below by writing `wrt LIST L`. We also specify that a list position is defined by its index, as well as conditions under which we can conclude an `LPOS` exists.

```
LPOS wrt LIST L
  i : INTEGER
  uniquely defined by i
  exists iff 0 <= i < L.length
```

In order to express the fact that a certain `INTEGER` is at a certain `LPOS`, we will also introduce a function `At` which maps an `LPOS` to an `INTEGER`. Because the process of ascertaining what integer is at a given `LPOS` involves sensorimotor behavior, we will for now simply write that the function has a "sensorimotor definition."

```
At : LPOS[L] -> INTEGER
  [sensorimotor definition]
```

By the way, in the treatment presented here, we are thinking of list positions in terms of integers, which makes sense given our earlier decision to have the agent access lists via numerical indices. But there are other ways to do things: we could have allowed the agent to "point at" positions deictically, to traverse from one position to the adjacent ones with "left" and "right" relationships, and so forth. One of the benefits of setting up the ontology in a way that distinguishes list positions from integers (which is what we have done) is that we leave the door open to later give the agent alternative ways of thinking about lists and their constituent positions.

Moving on, recall that in the previous section, we treated "occurrences of integers within a list" as entities. Now we come to the record for such entities:

```
INT-OCCURRENCE wrt LIST L
  position: LPOS[L]
  type:     INTEGER
  exists iff At(position) = type
```

Here `LPOS[L]` means "position in the list `L`."

We also need a type for the construct `(before (position 50))`. This is a "part of a list," which we'll call an `LPART`.

[28] I haven't tried to analyze what "sequence of" means in the definition here. I don't think it would be right to call it a `LIST` (even a different type of list) entity, because when one hears or says a number like "310" one would not say that one has provided a list.

[29] Obviously in speech we normally do not distinguish between the integer 3 and the digit 3; we just say "three." So we might wonder whether a similar stance would be appropriate in building representations. Should we really have to clarify which one we mean? To a large extent I don't think we can answer this yet, so I won't try.

[30] In our earlier TLOG for the list-searching task, we did something less sophisticated, which was to take integers as a primitive that could be exchanged freely between mind and environment.

```
LPLACE wrt LIST L
  pred: predicate over LPOS[L]
```

The `pred` field here holds (for any given `LPLACE`) a one-place predicate over positions in the list `L`. This is a slightly awkward definition, since it seems like not just *any* subset of a list could felicitously be called a "part," but we won't try to fix that now.

We have now finished describing the ontology we need. But before we move on, let's reflect on what exactly we have accomplished by writing down these concept records. Essentially, we haven't done anything we couldn't do with a bunch of first-order logic axioms. For example, when we said that an `INT-OCCURRENCE` exists iff `At(position) = type`, that could actually be formulated in first-order logic like so:

```
(forall L:LIST
  (forall p:LPOS[L]
    (forall n:INTEGER
      (iff (= (At p) n)                                              (11)
           (exists x:(INT-OCCURRENCE L)
              (and (= (property x 'position) p)
                   (= (property x 'type) n)))))))
```

But obviously our representation is much more compact! Moreover, assertions along the lines of (11), owing to the special role they play in ontology, surely receive some sort of special treatment in the mind, rather than merely being manipulated in exactly the same way we would manipulate an other logical proposition. So I think something is gained when we write ontological information in the form of concept records. Essentially we introduce a "layer of abstraction" above first-order logic. I suspect that such layers of abstraction are going to be necessary to accurately model human thought.

9.7.2 Interpreting the representation

Let's now see how our representation (10), reprinted below, can be interpreted in terms of the concepts described above.

```
(no-exist-in
  (occurrence-of One)                                               (12)
  (before (position Fifty)))[L]
```

I have made one change to the above representation, which is to include an "argument" `[L]` at the end of it, which will indicate that this whole statement is *about* the list `L`. We will assume that this annotation "propagates inwards" so that we know we are talking about occurrences of `One` *in the list* `L`, position 50 *in list* `L`, and so forth. The motivation for pulling `[L]` out of the main body of the proposition is that it is natural to suppress that argument in speech (as was done in (6)), so it seems like it deserves a special status.[31]

We can now explain the meaning of (12) from the inside out. First, `One` and `Fifty` are "known entities": particular instances of the `INTEGER` concept that the agent is already familiar with. We can notate (some of) our knowledge about them as follows:

```
One (INTEGER)
  digits: [1]

Fifty (INTEGER)
  digits: [5, 0]
```

Next, `(occurrence-of One)` denotes a "subtype" of `INT-OCCURRENCE[L]`s, namely the one containing instances whose `type` field is `One`. `(position Fifty)` denotes the unique `LPOS[L]` whose `i` field is `Fifty`.

We can define `before` as follows to map an `LPOS[L]` to an `LPLACE[L]` consisting of the positions before that one:

[31] Why is it so natural to suppress the argument that indicates what list we are talking about? It seems that it's because it's part of the "context" or "background," and can therefore be inferred easily.

```
before : LPOS[L] --> LPLACE[L]
(before pos) ::= (iota place:LPLACE[L]
                   (= place.pred (lambda (pos2)
                                   (< pos2.i pos.i))))
```

Here "`iota`" is the "definite description" operator ι commonly used in linguistics: $\iota x.f(x)$ means "the (unique) thing x such that $f(x)$." Other prepositions like `after` or `starting-at` would be similarly defined.

Finally, `no-exist-in` was already defined (see (9)), but it made reference to a predicate `in`. We can define `in` as follows, noting that `in` could simultaneously be given other definitions for other argument types (a situation analogous to polymorphism in object-oriented programming):

```
(in pos:LPOS[L] place:LPOS[L]) ::=
  (place.pred pos)
```

This simply checks whether the `pos` argument satisfies the predicate associated with `place`.

Having specified the above definitions and concept records, each term of (12) now has a well-defined type, and the statement as a whole has a well-defined meaning, in the sense that we can determine its truth value for any given `LIST L`. Our next goal will be to understand how the information contained in (12) is actually *used* to carry out the task.

9.8 Executing the task

We have finally figured out a representation for (6), but now how do we get the agent to use that information as it should? I will outline one proposal.

Basically we need to draw a distinction between *what* the agent wants to do and *how* it intends to accomplish it—let's call these the goal and the plan respectively. We assume that the agent uses reasoning to generate plans from goals, and once a plan is constructed, it executes with minimal thought.

Let's consider first the inference that needs to occur to get from goal (plus simplifying information) to plan, and then afterwards we will look at the control strategy that orchestrates when to draw what inferences and when to start executing a plan.

9.8.1 Inference

In the case at hand, the goal is to find a 1 in list `L`, the simplifying information is that there are no 1s before position 50, and the plan is to search starting at position 50. Here is a partially formal proposal for an inference rule to generate that plan:

$$\frac{\begin{array}{c}\alpha \text{ is a goal of the form } \texttt{(find x L)} \\ \beta \text{ is information about } \texttt{L} \text{ of the form } \texttt{(no-exist-in x y)[L]} \\ \texttt{(- whole-list y)[L]} \text{ has a relatively simple form}\end{array}}{\text{plan for } \alpha \text{ given } \beta = \texttt{(search x (- whole-list y))[L]}} \tag{13}$$

A few explanatory comments are in order here. First, note that we are computing a plan relative to some piece (or pieces, in a more refined version) of information β already singled out as being relevant. The reason for doing this, as opposed to something like (14), is that we don't want to assume that, at any given time, the agent is actually *aware* of all the pieces of information that it might know (or derive) about `L`. They might not have sprung to mind yet.

$$\frac{\begin{array}{c}\alpha \text{ is a goal of the form } \texttt{(find x L)} \\ \texttt{(no-exist-in x y)[L]} \\ \texttt{(- whole-list y)[L]} \text{ has a relatively simple form}\end{array}}{\text{plan for } \alpha = \texttt{(search x (- whole-list y))[L]}} \tag{14}$$

By annotating the plan with a variable β to represent what shortcut-enabling information has gone into it, we make it possible for the agent to later realize something else and come up with a better plan, without being in the awkward position of having to call them both "the plan for α."

Next, `(- whole-list y)[L]` is our notation for the complement of the region `y` in the list. The reason for insisting that it has a simple form is that otherwise it might not make sense to perform the complement operation. For example, if we are told that item 72 in the list is not a 1, then we might just ignore that information, or depending on the cost of checking items, we might be on the lookout for 72 so we can skip it. But we probably would not pre-compute a representation like "items 0 to 71 and items 73 to 99" as what we intend to search. (This illustrates a more general point, which is that human thought is "opportunistic": we will exploit all sorts of regularities when they are present, but we only use a given trick when it buys us something, not in the general case.)

Finally, `(search x p)[L]`, where `p` is an `LPLACE(L)`, has the obvious meaning of searching that part of the list `L`. We will see soon how that could be executed soon.

Now that we have the gist of the reasoning that should take place, let's look at the "control logic" that ties having a goal to the derivation and execution of the appropriate plan(s).

9.8.2 Control logic

I can imagine two types of situations in which I might find myself when given a goal like "find a 1 in this list" and some information about the list in question. First, I might *know what I know about* `L`—in other words, I might be able to say with certainty that "the only thing known about `L` is that there are no 1s before position 50." In that case, we use that bit of information as β in rule (13) and we can confidently work out the optimal thing to do. The second sort of situation we might face is one where we *don't* have a confident understanding of "everything that can be said about `L`." That could happen for various reasons: we might have been told some information about `L` but not remember exactly what it was (for humans at least, if we are told ten facts about something we are likely to forget some when asked to spontaneously recall them). Or we might have various pieces of information about `L` whose logical consequences we have not fully explored (and which do not currently seem worth trying to derive). Of course, in such a situation *some* relevant facts about `L` may come to mind. So it is those that we would want to plug in for β in (13), and then we would proceed with the resulting plan, pending something new occurring (or being told to us) that might merit revisiting the plan.

Actually, the above story should be refined in one small way. Whatever we information we have at hand regarding `L`, it is really only what I call the "discriminative" subset of that information that matters (and that we would want to plug into an inference rule like (13)). By "discriminative information," I mean information about `L` that would bear on what positions could potentially contain the target. So "no 1s before position 50" counts, but something like "all elements of `L` are odd" would not. What the agent will want[32] to know is what plan is appropriate in light of the discriminative information it has at hand.

In light of the above considerations, the control logic for proceeding with the goal might be something like the following:

- If your goal α has the form `(find x L)`, then attempt to retrieve discriminative information about `L`. (Note that this rule would typically be executed subconsciously.)
- Having done that, if the discriminative information β you were able to retrieve about `L` was exhaustive in the sense described earlier, derive "plan for α given β" and execute it as soon as it's derived.
- If your goal is as above and you have not yet been able to retrieve exhaustive discriminative information about `L`, then after some reasonable waiting period (that is, to allow memories and ideas to bubble up), derive "plan for α given β" (where β is whatever you have so far) and proceed with it.
- If new discriminative information comes up after the decision has been made to go ahead with β, pause what you're doing and reconsider the plan.

One extension worth noting: although in this case we got a single "right" answer as our plan, in other cases there might not be a single right answer (for example, there might be different levels of memory burden to save different amounts of list probes, or we might have to judge when to stop looking for shortcuts), and a decision about what to do might just be heuristic in nature. In cases like that, the conclusion of the inference rules playing the role of (13) would probably say that *a* reasonable plan (as opposed to the one right plan) is such-and-such.

[32] I am using the word "want" loosely here.

9.8.3 Mechanics

A couple final remarks on the execution of the process outlined above. First, how is `(- whole-list (before (position 50)))` actually evaluated? We assume that there is a rule like

```
       (- whole-list (before (position N)))
   --> (starting-at (position N))
```

which would convert `(- whole-list (before (position 50)))` to

```
   (starting-at (position 50))
```

as desired.

Second, once we have constructed our plan, what does its execution look like in the TLOG formalism? Perhaps something like this:

```
   L    ENTITY          List
   p    PLAN            (search One L (starting-at (position Fifty)))
   g1                   Execute(p)
   g2   GNSF       ^g1  Find(1, s); state = search
   i1   IT         ^g2  pos = 50
   x1   GNSF       ^i1  Probe(s, 50)
   ...
```

Here the goal of executing the plan sets up a `Find` goal node just like last time (see §9.3), leading to the same sort of serial searching behavior, but this time, the counter starts at 50. (This could be achieved by a rule which instantiates both `g2` and `i1` from `g1`.)

9.9 General comments

At this point we have a reasonably complete (if broad-brush) picture of how the extended list searching task could be performed. The picture painted so far is a good example of what I mean by a storyboard. It hits the highlights of the cognitive processing to be performed, but doesn't attempt to cover a wide range of possible tasks, nor to make all of the low-level formalization choices that would be required in an implemented system (it does make some though). The virtue of stopping here, at this intermediate level of detail, is that we can now move on to another task (or a variant on this one) and obtain a broader view of how an HLAI might function, rather than drilling into details that (a) can't be accurately filled in without that broader understanding (b) may need to be learned autonomously anyway.

Before we go on to a new set of tasks, a few closing remarks on this one. First, it bears mentioning that all of the concept records, behavioral rules, GNSF types, and other objects postulated throughout this section need some sort of origin story. The ultimate goal is not to create all these structures by hand but to figure out a way for a system to learn them autonomously. However, that goal of autonomous acquisition is still a long ways off. In fact we aren't really in a position to work on it yet, in large part because we need more data (that is, more storyboards) to use as a basis for theorizing. So for that reason we will shelve the topic for now.

Second, note that the behavior we have considered here (in the "extended" list-searching task) is an instance of a much broader class of desirable cognitive behaviors: reasoning from known information to save effort. We don't want our agent to blindly launch into any old procedure which should eventually work, but only to do the things that are necessary (insofar as it can discern what is necessary in a reasonable period of time). "Effort saving" enters the picture any time a goal is to be achieved which, if done in the most straightforward way, would be costly (in time, money, or whatever). One example of that phenomenon is tasks which involve iteration, as we saw here, but it is a much more general issue, especially in everyday reasoning (e.g. you *could* walk 10 miles to work each day, but you probably don't).

The list search problem considered here also illustrates our "middle road" (see §7) philosophy. If we were writing an ordinary computer program to search a list we would probably just "bash it" (i.e. use a straightforward "scan them all" strategy). It's only when we get to billions of elements that a brute force solution starts to look unattractive. So if we were to limit ourselves to writing code the way a programmer would, and simultaneously to tasks that a human could do by hand, we would find ourselves with no use for the sort of effort-saving reasoning discussed in this section. That would deprive us of an opportunity to model sophisticated thinking in a simple context. This is why we don't want to "just program." We want to take a middle road and include more cognitively plausible elements in our stories about how thought works.

10 The idea of an artificial scientist

As discussed in §7, a central part of the methodology we've adopted is *tasks*. The idea is to pick a task, analyze human reasoning as it appears in the context of that task, and reverse engineer from there. This raises the important question of what tasks are suitable. In the previous section, we got some mileage out of an extremely simple list-searching task that required a nominal amount of thinking. But obviously humans must deal with much richer types of tasks, and so must we.

Let me now propose one type of task that seems well-suited to our goals: building sensorimotor systems (or rather, figuring out *how* to build them). In other words, the task is to do "research" into how the world works, how to perceive it, and how to act in it. This type of "scientific" work is a good domain to study because it is suitably complex, has limited prerequisite knowledge, and also provides a future path forward on many other types of tasks.

In order to make scientific tasks tractable for our hypothetical AI (or artificial scientist, as you might call it), for now we will be assigning it extremely simple "research projects"—not the sort of complex work actual scientists do. For example, a possible project would be to figure out how to detect letters in small binary images (see Figure 3). The AS (artificial scientist) would be given a variety of training images and would try to work out general rules for predicting the letters contained in new ones. (Note for readers who are thinking that this sounds like a standard machine learning problem: although it does have that structure, the goal is *not* for the artificial scientist to be, or to construct, a classifier with typical ML techniques! More on that in a moment.) For lack of a better term, perhaps we can describe tasks of this sort as "microscience"—a sort of primitive, distant cousin of the work done by computer vision researchers or roboticists. So the idea is that we will observe the human mind at work on microscience and use that as a basis for developing ideas about HLAI, following the overall scheme laid out in §7.

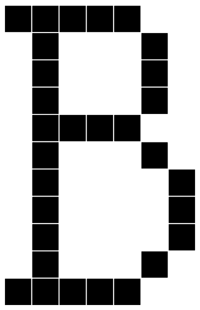

Figure 3: What letter is this?

There are two related arguments for using microscience as a target task. One is that in a certain sense, this type of work has some of the least prerequisite knowledge of any of the not-yet-automatable tasks humans can perform. This might seem an odd statement given that human scientists are highly educated, but one has to consider the vast amount of "common sense" knowledge that is required for other tasks but *not* for microscience. One can imagine a useful "microscientist" which knows very little outside of basic mathematics and how to develop theories. A microscientist needn't even understand the basic facts of the physical world (though it should work them out eventually), let alone complicated concepts like people, crime, or health. The same cannot be said of an artificial intelligence doing anything resembling the jobs of most knowledge workers! Nor would an artificial scientist need to start out with any particular sensorimotor capabilities (its whole purpose is to develop them), and again that cannot be said of most of the things humans do.

The minimal prerequisites for microscience mean that we can study how an HLAI might go about it without bogging down in questions about how the prior knowledge is to be represented or worrying that we have actually swept the hard problems under the rug by taking that prior knowledge for granted. In other words, microscience lets us start at the beginning, with concepts that don't depend on too many other complicated concepts.

The second reason why microscience is a good task to investigate is that, as discussed in §6.1.4, our long-term expectation is that HLAIs will acquire the capability to sense and act in the real world by building sensorimotor prostheses for themselves, and microscience is a microcosm of that process. The building of such prostheses is an important step in the development of HLAIs that can actually do more sophisticated tasks, including tasks of economic importance to humans. Not only would they allow HLAIs to do "manual labor" like cleaning or making food, they would (perhaps more importantly) provide a gateway for HLAIs to independently learn the myriad bits of common-sense knowledge that must be understood for more complex tasks. For example, to learn what it means for someone to "enter" a building, the easiest way is to watch someone do it (and be told that they're entering), and then deduce what it is about the geometry of the situation that is being referred to. This can only happen if one has access to the geometry of the situation, which is what a sensorimotor prosthesis provides. And of course, once one understands an idea like entering a building, one can understand more abstract ideas like going to work, deliveries, breaking and entering, locks, and so on. It seems likely that vast swaths of common sense knowledge are built up this way. Thus microscience is not only a convenient laboratory for studying thought, it is a preliminary investigation into a larger project that must ultimately be undertaken in order to provide HLAIs with the means to learn from the world.

The remainder of this report will focus on "microscientific" tasks and how a HLAI might go about them. In particular, we will look at a "microvision" problem, that of learning how to recognize letters in small binary images. (See Figure 3 for an example.) The letter recognition problem has already been intensively studied (for example in optical character recognition), but our interest lies more in the matter of how the scientific work can be conducted than in the problem itself. We will look at several aspects of this problem. In §11, we'll consider the process of acquiring templates for simple letters, both at a high level and in detail for certain steps. In §12, we'll look at a harder letter and some complications it introduces. Finally in section §13 we'll discuss the nature of concepts (a question raised in §5) in the context of our letter analysis problem.

11 The EFHILT task

In this section we will pursue the strategy advocated in section §7: we assign our hypothetical AS a task and analyze how it might carry it out. The task will be to recognize letters in small binary images, under a number of simplifying assumptions (starting with the assumption that we only care about the letters E, F, H, I, L, and T—the ones that consist of only horizontal and vertical strokes).

As alluded to earlier, the performance of any task can be analyzed at many levels of detail, from an informal overview to individual cognitive operations. We will discuss this task at several levels of detail. As one zooms in on a task analysis, more and more details become visible, and it becomes necessary to be selective regarding which parts of the picture to examine. Thus our more detailed modeling will not cover the whole task, but only parts of it. These parts nonetheless shed some light on what sort of cognitive architecture could work and what details must be filled in.

The EFHILT task is a simple one in the grand scheme of things, and in later tasks we will start to see some of the ways in which additional complexities enter the picture.

11.1 The task

As already mentioned, we want the agent to learn to classify images of the letters E, F, H, I, L, and T (with any given image containing exactly one of these letters). For simplicity, we will assume that the strokes are always one pixel wide and that the letter extends all the way to the boundary of the image. We will also assume we give the AS only "positive" data—that is, examples of letters of each type, but no examples of nonletters. For simplicity we are also going to assume we tell the AS that in principle image widths and heights can be anything.[33]

We have to be somewhat more precise about the desired end product of the learning process. First, there is the question of whether we want to learn enough about the letters to generate plausible new ones (i.e. a new "E" you haven't seen before), or whether it's good enough to merely learn to distinguish one letter from another. Let's borrow the terminology of machine learning and call models built for these two purposes generative and discriminative respectively. Building a discriminative model can be easier but less informative (for example in our case we can discriminate L's from non-L's by looking only at the upper-right pixel). Since we would like the AS to gain a "deeper" understanding of the letter forms rather than just finding simple tricks to disambiguate them, we will have it seek generative models for each letter.

In addition to the generative/discriminative distinction, we must ask what counts as a satisfactory representation for each letter class. And for that matter, where is the representation is to be found: in the mind of the AS, or in some external artifact (such as a computer program or diagram) created *by* the AS?

Since (at least in this simple case) there is no obvious need to complicate things by having the AS use external tools, we will assume that it builds its letter representations in its mind. In keeping with our view of the AS as carrying out a *deliberative* process of scientific investigation, these representations will of course be explicit (i.e. consciously accessible to the AS), which in turn means they will be largely symbolic, composed of chunks, and reasonably compact (due to human memory limitations, which we will respect until we have a reason not to). A desirable byproduct of having compact symbolic representations is that the AS would be able to convey what it has learned about letters in a way that is understandable.[34]

To summarize what we are after, we would like, for each letter, a simple characterization of what is typically[35] (or always) true of instances of that letter in our data set. We will be interested in both qualitative and quantitative assertions that can be made about our letter data (though our detailed modeling will focus on the qualitative aspects). At the risk of being a bit pedantic, our characterizations should have two additional properties:

1. Each characterization should be non-redundant (or at least not too redundant). Even if fact F is typically true of "L" images in our data, it should usually not be included in our description unless it can't be derived from other things in the description. So there's no need to say that "L"s have right angles when we've already said they have horizontal and vertical bars that meet.

2. Each characterization should include only simple propositions. A statement like "The shape is an exact match to one of $I_1, I_2, \ldots, I_n$" (where I_i is our ith sample image of the letter in question) is true but useless, and insisting on simplicity rules out such statements.

If we can get a characterization for each letter along the lines above, then it will allow us to do a variety of things: generate new plausible examples, recognize new examples of the letters (within the limits of generality of our data), and also extend our understanding of letter shapes to more enriched cases (e.g. where the stroke widths are not just 1).

What mode of characterization of letters actually meets the criteria outlined above? In other words, what sort of letter representations should we have our AS look for? I think we will want it to use "structural" descriptions of letters: descriptions of the subparts of each letter, their features, and their relationships. For example a "b" can be described as a vertical post on the left touching a circle on the bottom. This sort of description strikes me as the most robust and veridical way to capture our knowledge about letters, and descriptions of this sort are also amenable to construction without any fancy mathematical tools that we don't want to assume the AS knows about. So let's decide that it is descriptions of this nature that our AS will build. A good example of this sort of approach to letter recognition is Hofstadter and colleagues' Letter Spirit project (McGraw, 1995).[36]

Another question is the "format" we want our answer delivered in. Natural language? A formal description? UIL (see §6.1.4)? We will actually sweep this question under the rug for now, and simply assume we want the AS to build an internal, mental representation of what each letter is. The theory here is that that

[33]For now we will brush under the rug what happens if the image dimensions are unreasonably small (like 2×2).

[34]What would *non*-understandable knowledge of letters look like? Mainly here I am thinking of the "knowledge" produced by typical machine learning algorithms (e.g. neural networks or SVMs). There each letter class would take the form of a very large, difficult-to-interpret collection of numbers.

[35]Without trying to get too specific about this, let's say the cutoff for "typically" is something like 70-90% of the time.

[36]McGraw (1995) also contains a high-level discussion of possible approaches to letter recognition (see chapter 6).

is the main part of the battle; having the AS "read off" what it knows about each letter at the end is a comparatively small issue.

One other issue concerns the inductive biases we expect the AS to "bring to the table" regarding what sorts of regularities it should be looking for in letters. We have already partially addressed this question by stipulating that regularities should have compact descriptions, but it's worth saying a bit more.

As humans learn more and more categories of a given type (such as letters) and get more experience with each one, we learn more about what sorts of features do and don't matter.[37] This is why we can learn a new letter form (e.g. in a foreign language) with only one (or a few) examples: we know which aspects of that one instance are likely relevant and irrelevant. But an AS just starting to learn about letters would not have that prior knowledge. So for our purposes here in the EFHILT task, we will try not to assume *any* prior knowledge of what is likely to be relevant, at least not anything specifically related to letters. Obviously (by the no free lunch theorem) we need to assume *something*, but we are already doing that by seeking compact descriptions.

11.1.1 The nature of the agent's access to pixel data

There are many ways we could define a sensorimotor interface between the raw pixel data the AS needs and its core cognition. But for simplicity we will assume that the AS must probe the image by specifying the two coordinates of a pixel; its environment then returns the color of the pixel. (I call this the "battleship" method of interfacing with the environment.) This is analogous to the setup we used in §9.

11.1.2 A word on the visuospatial sketchpad

At least for humans, performing a task like the EFHILT task seems to involve the visuospatial sketchpad rather intimately—even if we assume (as we have assumed for the AS) that the image can be accessed only by the "battleship" method. When I do this task myself, at each step I use the VSSP to maintain the current situation, and at the end when constructing a verbal description, I "read it off" the VSSP.

However, since we have not made any proposals about what a visuospatial sketchpad would look like for an artificial agent, our models of the EFHILT task will not make use of such a feature. The AS's behavior will therefore not be directly comparable to what a human would do. However, we anticipate that it may eventually be possible to recast the representations we will propose in such a way that they play a role analogous to VSSP content.

11.2 An informal synopsis of behavior

We'll now start our storyboards of the AS's behavior on the EFHILT task. Let's begin with a general sketch of how it could go about the task. The objective is essentially for it to develop a "template" for each letter class.[38,39] The natural way to start would be to focus on one letter class, and within that one single image.

11.2.1 Developing a letter template

Suppose the AS begins by examining the image of an "F" in Figure 4. First it observes that the top row is all black. Having noticed something interesting, it could either immediately compare another example of the same letter (to see whether it's true there too), or it could continue learning more about this one; let's arbitrarily take the latter route.

The AS next notes that the left column is all black, and then (after scanning the rest of the image) that the only other two black pixels are next to each other at $(1, 3)$ and $(2, 3)$. These are remembered as a pair due to their proximity to each other and the fact that they are the "leftovers" once the larger structures have been identified.

[37] E.g. topology matters a lot like (think "t" vs. "T"), but as long as relative positions the same, particular lengths are not too important. Also in noisy real world data, single isolated pixels not too important (they can be often be ignored), nor are perfectly straight lines.

[38] For our purposes a "template" will mean a structural description of the letter in terms of parts and relationships (as discussed earlier); this is different from the way the term "template" is used in the psychological literature.

[39] The notion of a "template" may well be implicit, not known as such to the AS.

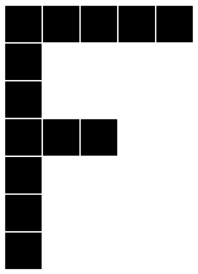

Figure 4: One of the "F"s the AS will investigate.

At this point we have what I call an "account" of the image: a short description of all the black pixels.[40] Here is how the account might be verbalized:

> The black pixels in this figure are the top row, the left column, and the pair $(1, 3)$ and $(2, 3)$.

(And although it is not stated explicitly, the fact that these two pixels have the same y coordinate and are adjacent would not escape the AS. It's just that repeating the "3" is a more concise verbalization than anything that points out the repetition explicitly.) The account allows the image to be precisely reproduced, but is much more compact than the image itself.[41] By generating that account the AS is now in a position to look at other "F" images and compare them to this one.

We now move on to another "F"—the one in Figure 5, let's say. This time, the AS again observes

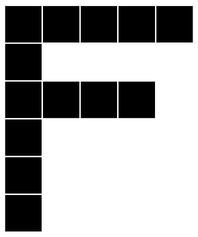

Figure 5: Another "F" image.

that the top row and the left column are all black, and it notes the parallel to the previous instance. Also like before, there are a small number of black pixels left over. It can now realize that these have a simple characterization: they are what I'll call an HBAR: a group of consecutive pixels in a single row.[42] (It might not have done so in the previous image because the HBAR notion would be less salient when there are only two pixels. But at this point it would likely notice that the HBAR characterization applies to the leftover pixels in the first example too.)

The AS now sees that both "F"s have generated very similar accounts. Each "F" consisted of the top row of pixels, the left column, and an HBAR (which in both cases began just right of the left column). The other properties determining the exact set of pixels included in the HBAR (namely its vertical position and width) varied by image, so they would be dropped from the template, but retained for later (see §11.2.2). At this point the AS might be prepared to venture a hypothesis, which if verbalized would be something

[40] That it makes sense to produce an "account" is a bit of "scientific" methodology that may well need to be taught (not necessarily explicitly!). Like language, it is a piece of "culture," that body of ideas we have developed as a species and pass on from one generation to the next. By the way, the notion of an "account" is applicable across all sorts of investigative tasks. Consider the problem of describing the numbers in the set $\{3, 6, 9, 12, 15, 16, 18, 21, 24\}$ (multiples of 3 up to 24, plus 16) or even of describing the conditions under which a mechanical part might fail as "when the temperature is over 140° F or force over 600 N."

[41] One can also envision accounts that only allow partial reconstruction.

[42] I use the SMALLCAPS font to indicate a specific concept the AS has in its mind (as opposed to one that only exists in *our* minds as we characterize what the AS is doing).

along the lines of (15).[43]

> I suspect that an “F” is a shape consisting of the top row of pixels, the left column of pixels, and an HBAR starting at $x = 1$. (15)

(Never mind that HBAR is not a word but rather a concept, and may not have a particular name—just insert some appropriate circumlocution there.) So now the AS knows that both images consist of the left column, top row, and another HBAR with its left end at $x = 1$. These middle bars do differ in their other properties (length, right end, and vertical position) so it notes that as well.

The AS now checks more “F”s against this understanding to validate its correctness. After seeing a few more that match, it has some confidence in its description.

So far, we have a qualitative model of “F”s: a description of some properties that all of them seem to satisfy. But note that our description does not fully determine how to draw an “F”, because there were properties of the HBAR that varied across instances. (Note that our description also doesn’t determine w or h, but that’s fine because we were told they could be arbitrary.)

There are a couple of things that in principle *could* be going on in a situation like this. One is that there is actually a further qualitative constraint (e.g. on parts and their connectivity and relative position) that we just haven’t found yet. Another is that the indeterminate variables for the crossbar of the “F” are truly free. But another possibility (the real one in this case) is that there exist constraints on the crossbar that are quantitative, which typically also means that there is a whole range of possibilities and its boundaries are fuzzy. For instance in an “F” the HBAR is usually around the middle, but there is no exact point at which we can agree it ceases to be “F” (or even starts to be atypical).

Our AS should theoretically learn about these constraints as well. In our case, the agent would reach for quantitative tools after establishing that there is no obvious qualitative constraint (within its “known qualitative constraint types”) for pinning down the location of the crossbar of an “F”. Since quantitative constraints are a more complex matter than qualitative ones, in our cognitive modeling we won’t tackle them in any detail. (Generating a characterization of sample data from a two-dimensional space is on the same level of complexity as generating a characterization of a two-dimensional image.) However, let us at least say a few words about how it would be done, at a high level.

11.2.2 Quantitative investigation

First, we have a choice as to whether to gather data about the “F” shapes and analyze it later, or whether to do it as we go. Either way, the basic task is to characterize what combinations of numerical parameters are typical for “F”s. (In the case of an “F” we have four parameters to deal with: the width and height of the image, which we already know are arbitrary, and the elevation and width of the crossbar.)

This characterization is likely to involve standard methods of descriptive statistics: means, correlations, regression lines, and so forth. (On more complicated problems, an AS could conceivably draw on *anything* from the machine learning/statistics arsenal, but we need to keep things simple if we are to have any hope of formally modeling the knowledge of the AS.)

How far should the AS go in attempting to find patterns in the data? This is something of a deep question (relating to the well-known bias-variance tradeoff), and we will not delve into it except to say that in simple cases like this one, (mathematically inclined) people tend to have some intuitive sense of what is reasonable. For example, on a data set of 15 “F”s, I came up with the following characterization which seems like a reasonable level of detail (here w, h, x, and y represent image width, image height, crossbar width, and crossbar elevation respectively):

- 15 data points examined
- w ranged from 3 to 7, h from 5 to 10, not particularly correlated
- x is typically close to w: w or $w - 1$ 14 out of 15 times

[43] Another thing to be pointed out about nascent hypotheses is that they are not unitary: if we were to get another case that matched the “F” pattern in all respects except the left end of the HBAR, we’d be likely to (tentatively) extend our concept to cover it. In general, hypotheses are not discarded if proven wrong but tweaked. Think of a hypothesis not as a monolithic object but as more of a generality that is shared by a group of data points seen in the past. When new data comes in, that generality can be adjusted. See §13 for further discussion.

- y is typically about $(h-1)/2$: within ± 1 on all data points

"How far to go" in terms of trying to extract patterns from the data (and for that matter how much data to look at when many training instances are available) is something that could potentially be learned heuristically, from experience.[44] Certainly it seems this is the way humans do things; for instance, to be satisfied with the qualitative model of "F"s discussed above, I would be satisfied with it working for the first 10 or so examples, even if more were available. At that point it makes sense to move on to the next task, at least if one is "just exploring." (If there is something riding on being sure one is right, then the calculus changes of course.) From a modeling perspective, when we don't have a heuristic for this sort of thing at hand, we can just leave it blank and get a nondeterministic model, which is fine.

A few small comments on the above "statistical report":

- Why describe x and y in terms of w and h rather than other way around? This seems intuitively reasonable for various reasons. For instance, w and h are more "fundamental," since they can be immediately computed as soon as you have the image, without doing any work on it.

- It's important to not just characterize the empirical pdf[45] of a set of data, but also know something about how much data it's based on, so that we can make judgments later about how confident to be. In the same vein, we want to know what range of w and h we've looked at, because even though we're told these can be anything, we want to be careful about extrapolating beyond the range we've seen data for.

- This is a good case of a vague term like "close" coexisting with a more fine-grained explanation of what "close" means in the relevant context. Why even bother with the word "close" when you're just going to supply the details? Because we want to be able to reason at multiple levels of detail. To give just one example of how this could be useful, "close" generalizes to the continuous case whereas w or $w-1$ does not.

- Often you can frame something multiple ways (w or $w-1$ could also have been described as "near $w+.5$"). Usually we want to avoid redundancy, so choosing a description is a matter of optimizing a combination of informativeness and description length.

11.2.3 Wrapping up

Going back to our story, once the AS has finished (to its satisfaction) creating its collection of qualitative and quantitative rules characterizing "F"s, it can move on to each other letter in turn and do something similar. For each of the EFHILT letters, a characterization along the same lines (that is, one involving full rows, full columns, and in a couple cases HBARs) is possible, so nothing particularly different comes up with any of the other letters. Once it finishes with all the letters, the AS is able to identify a new letter upon request, and also report on its findings as to what each letter looks like. The AS is thus done with the EFHILT task.

We have now seen a high-level overview of the EFHILT task. The next task is to turn up the magnification and devise specific representations and mechanisms that could achieve the effects described in this overview. We will start doing that soon, after we fill in a few details about the intended behavior of our AS that were not clear from the single example discussed so far.

11.3 Some general remarks on account-building

Since "account building" was the central activity in the foregoing analysis, let us look closer at accounts. After that we will move on to a more granular analysis of the account building process.

First, obviously, accounts are not unique; one image could have many reasonable accounts. One way this can happen is that parts meet or overlap, as they do in an "F" shape. One then has multiple choices about which account-parts to assign the common pixels to. Should the upper left pixel be seen as part of the top horizontal bar, the left vertical bar, or both? There are several ways to handle this sort of ambiguity.

[44] I.e. the AS would somehow receive reinforcement as a result of underfitting/overfitting, or overexploring vs. underexploring the data (due to an investigation time vs. accuracy tradeoff).

[45] Probability density function.

First, one could establish some conventions about which accounts to prefer. Second, one could have lenient templates that accept any of the possible analyses. Third, one could say that an image matches a template if *any* of its possible accounts matches the template. Any of these approaches is legitimate, but the problem must be addressed one way or another (and we will see it show up in §12).

Another important aspect of account-building (displayed by humans and necessary for the AS too) is to produce *minimal* accounts—that is, accounts that would be incomplete if any part were removed—and to simplify accounts that for whatever reason are not minimal. For example, we would not want to say that a figure like Figure 6 consists of "the top row, the left column, and an HBAR" (counting the top row as an HBAR). Part of the meaning of "consists" is that the parts are distinct; it is not just a synonym for set-theoretic union. In fact, it seems that "consists" means something even stronger: we need the account

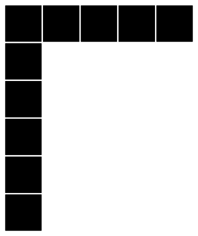

Figure 6: An F?

to be a *natural* one. The parts must each have some sort of coherence and the account must not have some trivial simplification.[46]

It is hard to state precisely what makes a natural account (and therefore what "consists" means), but we humans "know it when we see it." In other words, to some extent our knowledge of this concept (and many others for that matter) is implicit.[47] Relatedly, when we *use* the term "consists", either to describe an account we have generated or to confirm that someone else's account is a good one for a given image, we are not reasoning from the definition of "consists", but instead carrying out some procedure.[48] "Consists" thus has a procedural aspect. One can say the same of a conclusion like "An 'F' has a Y such that Z." This is generated not so much from a *definition* of having a part as from a procedure that directly builds such propositions from the data. Our AS should probably function in a similar manner.

11.3.1 The implicit nature of concepts

By the way, the "procedural" and/or "implicit" nature of concepts does have one interesting implication, which is that thought is not necessarily built upon a set of rigid axioms and definitions (indeed, trying to build it that way is one of the problems with traditional "logical" approaches to AI). Concepts seem to exist in a more approximate form, one involving rules about what does and doesn't count. These rules needn't cover all circumstances, and could even contradict each other in edge cases that haven't come up yet.

The implicit nature of concepts may also explain the gulf that sometimes exists between intuition and formalization, for example in mathematics and in attempts to formalize various other domains of knowledge. When we start to formalize our concepts, we can discover that our formalizations lead to counterintuitive conclusions, because the rules underlying our intuition were in fact contradictory, or because we weren't aware of certain aspects of our intuitive understanding when we attempted to generate the formalizations. This sort of situation is what led to Russell's Paradox, for example. (Other similar quandaries in math include the difficulty of proving the Jordan Curve Theorem, the Banach-Tarski paradox, the fact that one cannot meaningfully assign an area to arbitrary subsets of the plane, and even the Monty Hall problem.) Similarly, this phenomenon is responsible for the nagging feeling of unease that accompanies just about *any*

[46]Consider for example a case where "F"s did not need to extend to the bottom of the image—one might then try to circumvent the non-redundancy requirement by using a 1-pixel HBAR for the bottom pixel and saying the VBAR ran to just above it. But such an account would not be trivially simplifiable and therefore unnatural.

[47]It is not *entirely* implicit though, because when X does not consist of Y we can generally explain why.

[48]What procedure exactly? Well to generate an account we have a loop like "while content unaccounted for, find a good 'part' and add it to the account being built."

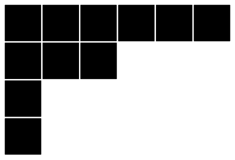

Figure 7: Not a good "F", but the AS wouldn't know that at first.

attempt to formalize some piece of common sense knowledge: one can't help but doubt that one has actually formalized the relevant concepts in a way that would properly reproduce human judgments. (This is the reason I view "theoretical" efforts to formalize knowledge—that is, efforts that build knowledge without seriously exercising it in service of performing tasks—as a questionable research methodology: one has the feeling the formalization will prove far off base in a way that would become obvious if one considered a broad range of actual tasks.)

All of the above being said, there are sometimes reasons to formalize concepts—for example, making specific rules how we will conduct image analysis and when we will make certain determinations helps us to conduct controlled experiments about vision architectures. Moreover, if we want to *prove* things about letter forms (e.g. prove that something does or doesn't match a template or that some system will detect all matches to some template) we need to formalize. Somewhat further afield, formalization of concepts, in the sense of making them more clear than they would otherwise be, plays a role in matters of law and policy.

We will return to the matters discussed here in §13.

11.3.2 Reification

Throughout our discussion of account-building we have used terms like "template", "account", "constraint", and "role". I think it's important to point out that the agent doesn't necessarily need to have any of these notions, at least in the sense of having a sortal concept that corresponds to any of the above nouns. Another way to say this is that while the AS would "implicitly" use the notion of accounts anytime it built an assertion "X consists of Y", it needn't have the *reified* notion of an account. This means among other things that it wouldn't be able to *predicate* accounts, templates, etc.

11.3.3 Vision-specific preferences on accounts

We could, on top of the "basic" rules for accounts (that they should be comprehensive and non-redundant), learn more specific rules appropriate to visual (or even more specifically, letter) accounts. For example, a general rule for analyzing images is that sets of pixels chosen as "parts" should have their edges at "natural" boundaries—boundaries that exist in the image. To see what I mean, consider a shape like Figure 7. It would be awkward to treat the top row as one part and the second row as another part (and thereby identify this as an "F"), because the boundaries of these two parts do not have sufficient alignment with black-white boundaries in the image. But we will not try to model this preference for now, nor shall we assume the AS would know it at first.

11.4 More specific models of account-building

As promised, we will now start to get more precise about the account-building process sketched earlier. This process has a simple overarching structure: the agent goes through the pixels of the image (not necessarily in any predetermined order) and keeps track of what has been done so far and what has been learned. I call these two pieces of information the "state of work" (SOW). Concretely, the SOW should represent what pixels have been explored and which ones were black. (These sets of pixels should be represented at a higher level than just a list of pixels of course, for example by being grouped into HBARS and VBARS.) When all pixels have been explored, the scanning process is complete.

To specify this process in more detail, the main questions to be answered are these:

1. How does the agent represent the SOW?

2. How does it decide which pixel to check next?

We will propose some answers to these questions in this section. A third question concerns the actual implementation details for the overall strategy we discuss; we'll look at (parts of) that issue in §11.5 and §11.6.

11.4.1 Objectives in modeling account building

Before we get to specific proposals about representations and decision mechanisms, let's take a moment to ask what we want out of our model. Obviously a model of account building can be built in many different ways. So what desiderata constrain us? Are we just trying to get a system that works? Are we trying to figure out exactly what humans do? As discussed earlier in section §7.3, I want to take a "middle road": the goal is behavior that is human*like*, but without trying to uncover *exactly* what a human might do, a goal for which we have no adequate experimental methodology. In other words, we want partial but not total veridicality. Let me illustrate with an example.

First consider how we might build a program to do the account-building task if we did not care at all about being faithful to human behavior. The pixels could be scanned row by row, left to right. Representing the current position in the scan could be achieved simply by storing the (x, y) coordinates of the last pixel probed. Black shapes found so far could be represented as a collection of HBARs, VBARs, and isolated pixels. Any new pixel that could merge with an existing item to form a larger one would do so, and if not it would become its own pixel. Each individual HBAR or VBAR could be represented with three integers for its position and size. The final account would be whatever set of items existed in the collection after scanning all pixels.

This program would be easy to write, but doing so wouldn't teach us anything about intelligence. Looking at the above description, we can identify several respects in which it is unlike what a human would do. I think there are two main differences. First, a human would not necessarily explore pixels in a simple, fixed order. Second, a human would use a richer representational repertoire. For example, instead of representing everything with numbers, she would probably think of (0,0) as being the "upper left corner". Instead of just HBARs and VBARs she might use higher-level structures like "L shapes". She might avail herself of the opportunity to describe the SOW in terms of known-black and known-white pixels, if that were simpler than specifying scanned pixels and black pixels. And so on. Also observe that a human will remember (some) information about SOWs prior to the current one, but presumably not by copying the entire SOW data structure each time it changes.[49] This suggests a versioned data structure. We want features like these to be reflected in our models. So in studying representation we are not really interested in capturing "just the facts". We also care about how they are organized.

Our goal will be to create a model that incorporates some of the features outlined above, without claiming to do *exactly* what a human does (whatever that would mean) or to capture every possibility. In building this model I have relied mostly on my own introspective observations while pursuing the task. Undoubtedly the particular sequence of states I have storyboarded would be different for any other subject (or on any other day), but the important point is not to capture *every* possible variation. We just need to capture enough of them that we can generalize in the future.

Now that we understand what we are trying to do, let's discuss how the AS might control its scan and track its progress. We'll start with the first issue.

11.4.2 Heuristics for image scanning

How do humans decide what pixel to look at next? I'll discuss some general heuristics that seem to be operative here, relying on my own introspection. In general, the gist is to choose the scanning order in such a way that the information to be remembered is kept simple and compact.

There are two aspects to decisions about the scan order: what factors determine those decisions and how the decisions are actually made. We'll mainly focus on the first question. I believe this can be reasonably well-modeled with a set of rules like the ones below. (These are general ideas; I'm not claiming they're exactly right!) When we go through an example in the next section, you can check that the order of exploration is consistent with these heuristics.

[49] That would not be consistent with the limitations on human memory.

1. If possible, pick something adjacent to the black pixels already identified.
2. Prefer a pixel which, if it turns out to be black, would cause the total black region to have minimal complexity. For example, in the situation in Figure 8, it makes more sense to probe pixel A next than pixel B, because if pixel A is black, we will have a black figure (a rectangle) that can be described compactly, but if pixel B is black we get a complicated one.

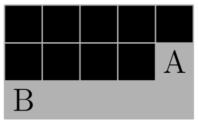

Figure 8: If the black pixels have been probed so far, better to probe A rather than B next.

3. Prefer pixels such that the new scanned region will have a simple form. (It's possible for this rule to be in conflict with the previous one, for example in Figure 9; we won't worry about this for now.)

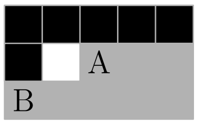

Figure 9: A conflict between two heuristics for what pixel to probe next. Picking pixel A would result in a simpler scanned region, but pixel B (if it's black) would yield a simpler black region.

4. Prefer pixels that have more contact (in terms of number of sides) with the existing black region, and/or more contact with the scanned region (but not at the expense of rules 2 and 3).
5. If starting, pick $(0, 0)$.
6. If you have a bar (2 or more pixels in a row), consider extending it.
7. (Mildly) prefer "extreme" pixels (i.e. highest, leftmost, etc.).
8. If you find yourself following some sort of systematic procedure, like going row by row, mildly prefer to continue going row-wise rather than switching to column-wise.

It might be worth asking at this juncture *why* humans don't just always scan the pixels in some simple fixed order; when considering letters like we have done, this would seem not to create any great problems. But consider what would happen if we tried to apply that rule when reading a whole page of text. We would have to store quite a lot of information (well beyond what a human would find convenient) from the first few rows of pixels before assembling larger structures (i.e. whole letters) that could be more compactly stored.

To return to the other point we raised at the start of this section, what is the *method* by which these decisions are made? I don't want to get into a detailed discussion of how the heuristics might be computed, but only to point out that often we probably don't "redo" the decision process for each pixel. Instead, if you have begun scanning a row and it has been all black so far, and there is no particular reason why (assuming the row continues to be black) you wouldn't just keep going, you will probably just keep advancing within that row without "redoing" the decision process. So when we try to model how the mind does the task, we should make a provision not just for a heuristic decision process, but for creating and executing "intentions" that control behavior over multiple probes.

The kinds of ideas above may apply more broadly to all sorts of investigations we might want a microscientist to undertake. In general, humans seem to preferentially investigate questions that allow us to keep the facts we are tracking neat and tidy, and we pursue any pattern that might start to emerge. This is not just some sophisticated heuristic restricted to laboratory experiments by professional scientists: it is something that applies even on mundane problems like the EFHILT task.

11.4.3 Representation of SOWs

Now we get to the question of representation: how might a "middle road" AS represent its state of work on the EFHILT task? We will develop our proposed representational formalism by going through an example, using the "F" shape of Figure 4.

Our representation is based on the idea of a "chunk", a mental data structure which has a collection of fields and values. This terminology follows ACT-R (see §6.3.2), but a chunk is more or less the same thing as a "structure" in a typical programming language or an "attribute value matrix" as used in certain approaches to linguistics. As a notational convention, we write a chunk either as

```
ChunkType(field1: value1, field2: value2, ...)
```

or as

```
ChunkType
  field1: value1
  field2: value2
  ...
```

Note that a value can be another chunk. We do not insist that all chunks of a given chunk type have the same complement of fields.

Our first few SOWs are shown in Figure 10. We will build a representation for each in turn.

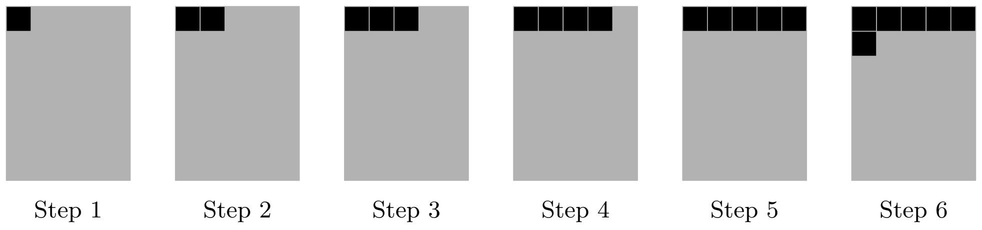

Figure 10: First few states of work while scanning an "F".

Step 1. Let's represent the situation so far by saying that there is a single item, namely a black pixel in the upper left corner:

```
SOW
  sole-item: Pixel
               where: upper-left
               color: black
```

Step 2. Our item now becomes two pixels; treating that as its own chunk type seems reasonable.

```
SOW
  sole-item: Two-Pixels-Horiz
               where: upper-left
               color: black
```

Step 3. Something similar happens again, but this time we introduce a width parameter:

```
SOW
  sole-item: HBar
               where: upper-left                                          (16)
               width: 3
               color: black
```

Noting that a certain amount of structure has been repeated in these first three representations, we speculate that the AS should have the ability to "factor out" what is common among multiple SOW structures. This can be done by introducing a "template" chunk structure where some of the fields have been variabilized:

```
ABSOW(x) ::=
  SOW
    sole-item: x with
                  color: black
```

Here a template `ABSOW(x)` is being defined such that for any given chunk structure A, `ABSOW(A)` represents the chunk structure obtained by substituting A where x appears in the definition. Field values mentioned in the template (just `color` in the above example) are to be merged into whatever chunk fills the x position. Thus (16) can now be represented as (17).

```
ABSOW(HBar(width: 3, where: upper-left))                        (17)
```

The careful reader will note that I could have chosen to fold "`where: upper-left`" into `ABSOW`, but I didn't. The reason is that introspectively, it seems to me that whenever I think of these SOWs, I visualize the shape on my visuospatial sketchpad, and when I do so, I inevitably see both the shape and its location. Thus it seems to me that it would be odd to create a chunk representing one of these shapes without mentioning its position. But if the position is always mentioned, there is no point putting it in the template.

By the way, the representation we have used here is by no means the only reasonable one. For example we could also do something like (18).

```
SOW
  explored: HBar(width: 3, where: upper-left)₁                  (18)
  findings: all(black)([1])
```

(The 1 subscript here serves to tag a part of the structure for later reuse; $\boxed{1}$ refers back to that part.) Interestingly, we can abstract out the recurring parts of this structure using definition (19), which puts us back in the same place we were in with (17).

```
ABSOW(x) ::= SOW
               explored: x                                      (19)
               findings: all(black)(x)
```

Step 4. Moving on to step 4, we can see that it has almost the same structure as the prior step, so we might even go further and define

```
HABSOW(x) ::= ABSOW(HBar(width: x, where: upper-left))
```

Then step 4's structure could be represented as

```
HABSOW(4).
```

By the way, I want to admit the possibility that such abstractions could be built "on the fly" whenever an opportunity is noticed, and that they might be short-lived, which would make sense for excessively specific ones like this. Of course, longer-lived abstractions should also be possible.

Step 5. At step 5 we would have something like

```
ABSOW(FullRow(row: top)).
```

Step 6. This step is similar to the last few; we just introduce a new type of shape. We use `UpperLeft-LShape` for an "L" shape oriented with the corner in the upper left.

```
ABSOW(UpperLeft-LShape(width: full, height: 2, corner: upper-left))
```

Step 7. At this point, the agent will presumably be issuing requests to probe pixel $(0, 2)$, $(0, 3)$, and so on; rather than adding one each time to obtain the height of the "L" shape, we will just store its bottom coordinate, like so:

```
ABSOW(UpperLeft-LShape(width: full, bottom: 2α, corner: upper-left))₂
```

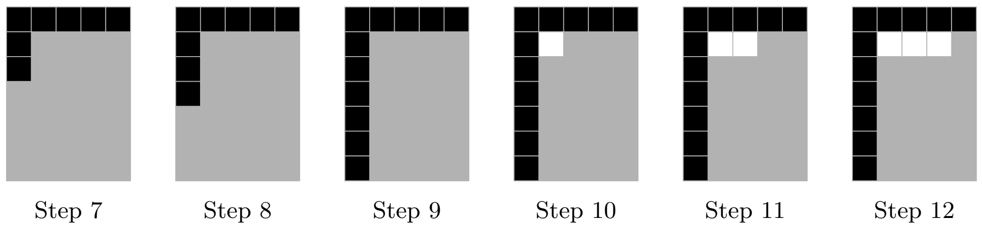

Figure 11: More states of work.

We have again used a numeric tag here, and we have also added a Greek subscript. The difference between numeric and Greek subscripts is that numeric subscripts tag the structures they follow, whereas Greek subscripts tag a *role* within a structure. So α here means "the value of the `bottom` field"; it is not tagging the number 2. Both types of tags are assumed to exist in the agent's mind; they are not just for our notational convenience. We will see the use of these structures momentarily.

Step 8. At this point, the structure is very much like the structure on the previous step. A prior situation like this one led us to introduce a "template" chunk structure; here we will achieve the same effect without explicitly constructing any template. Instead we will just declare that the new structure is like a previous one, with a certain substitution being made. In this case, the number 3 will occupy the α position, instead of 2.

$$\boxed{2}\,[\alpha \to 3] \qquad (20)$$

This sort of representation can be seen as storing a "diff" between the structure at hand and a prior one. Introspectively, I think this is consistent with the phenomenology; after probing the new pixel I would not mentally review the entire set of facts known so far but would probably just say to myself "3" or "bottom 3 now". (Periodically we do tend to review everything learned so far; this helps us retain the information and perhaps makes it easier to draw inferences from. We just don't do it on every step.)

I won't take a position on whether, psychologically, this idea should be regarded as psychologically distinct from the templates discussed earlier, though if there is a difference, it may be related to the level of transience of the template.

Step 9. Here we are back to writing out the SOW in full rather than as a diff like (20), because it seems more efficient to express the shape by saying it has "full" height instead of storing a number, but $\boxed{2}$ does not have a `height` field. (Perhaps with some tweaks to the formalism we could work around this, but we won't pursue that idea here.)

$$\texttt{ABSOW(UpperLeft-LShape(width: full, height: full, corner: upper-left)}_3\texttt{)}$$

Step 10. Now that we have begun to find white pixels in addition to black ones, we switch to a different way of specifying SOWs, namely by listing black and white parts. Now that we have the "L" shape tagged as $\boxed{3}$, we can refer to it in building the chunk structure:

```
SOW$_4$
  black: $\boxed{3}$
  white: Pixel(pos: $\boxed{3}$.innercorner$_5$)$_\beta$.
```

Here we also introduce the usage of the notation `x.y` to mean "the `y` of `x`."

Step 11. Here we apply our diff notation again, which results in a pleasantly compact representation.

```
$\boxed{4}$[$\beta \to$ HBar
          left: $\boxed{5}$
          width: 2]
```

For the sake of variety, let's also explore a second way to represent this same SOW. Consider how we might explain the SOW verbally:

> We have found a black "L" shape covering the top row and left column. In the remaining rectangle, all we have so far is a white HBAR of length 2 in the upper left. (21)

We can reflect this way of thinking about the SOW in the following structure.

```
SOW$_6$
  info: [ all(black)($\boxed{3}$),
          in $\boxed{3}$.complement,
            sow = AWSOW(HBar
                          where: upper-left
                          width: 2) ]
```
(22)

A few notes on the notation. First, the `info` field indicates that we are explaining the SOW by means of supplying a sequence of assertions, rather than filling in particular bits of information that answer particular questions.[50] The sequence itself is notated with square brackets. `all(black)(`S`)` means that the shape S is all black.[51] The notation "`in` R`,`" is meant to set up a context such that whatever is in the scope of that context (for example the tag `upper-left`) is interpreted relative to the region R, rather than the image as a whole. "`sow =` O" is the assertion that the state of work (relative to the current space) is given by the chunk structure O. Finally, `AWSOW` is defined the same way as `ABSOW`, but with white instead of black.

I will continue to use structure (22) for future steps, although this choice is basically arbitrary. I don't think we can say with any confidence which style the human mind prefers, or more to the point, which would make more sense in the context of an AS. This phenomenon—being able to say the same thing in many ways with no obvious right one—is pervasive when trying to write down representations, and we shouldn't let it trouble us too much. If some representational choices that seem reasonable now are actually problematic, then that should become clear as we move towards the formalization of the mental procedures that manipulate these representations, and ultimately towards working systems. We don't need to tie ourselves in knots now trying to decide which representation is "better". We should merely try to get a sense of the variety of reasonable options (while rejecting things that are not cognitively plausible).

One more comment on structure $\boxed{6}$ (and the English gloss (21) it was inspired by): in some sense it feels quite *natural* for the assertion `all(black)(`$\boxed{3}$`)` to be followed by an assertion which begins by narrowing its scope to the remaining rectangle (a linguist would say that said rectangle has been *topicalized*). Why is this? I suspect it is because when we hear a SOW described to us (or when we retrieve it from memory, which is basically the same thing for our purposes[52]), we are tracking, as we listen, whether we have received a "well-formed" description of a SOW. By "well-formed" I mean that the information we are given is non-contradictory, non-redundant, and fully describes a SOW. To do this, we will often keep track of "what remains to be learned" about the SOW. In this case, after hearing "`all(black)(`$\boxed{3}$`)`", the remaining information to be supplied is what is going on in $\boxed{3}$`.complement`: which pixels are black, white, or unknown. Then when we *hear* "`in` $\boxed{3}$`.complement,`", we are no longer separately tracking the region we need information about and the context that currently applies to the incoming assertions: they have been brought into alignment and the memory burden is reduced. This tendency to "make things easier for the listener" is reflected in the feeling that to topicalize $\boxed{3}$`.complement` is "natural". In fact, we might conjecture that it does not take much effort for the human mind to store the "`in` $\boxed{3}$`.complement,`" structure in LTM: it only needs to remember that the natural question to ask is getting answered.

I cannot resist one final remark: introspectively, it seems to me that when I retrieve $\boxed{6}$ (which, as mentioned earlier, is accompanied by visualization of the situation using the visuospatial sketchpad) and get to the `in` $\boxed{3}$`.complement,`" part, I "adjust my attention" so that I am now "looking at" $\boxed{3}$`.complement`. In other words, the "`in`" construct correlates to a VSSP usage pattern.

Step 12. This is another case where it makes sense to use a diff. I will repeat the chunk structure from the previous step for reference, this time with some indices added:

[50] For example, we previously used fields like `sole-item` and `black` to answer the questions "What is the one item found so far?" and "What pixels in this image are black?" respectively.

[51] Formally, `all` is a function whose argument is a color and whose output is a one-place predicate over shapes.

[52] I am supposing that when accessing it we would generally traverse the elements of the `info` list in order.

```
SOW₆
  info: [ all(black)([3]),
          in [3].complement,
            sow = AWSOW(HBar_γ
                         where: upper-left
                         width: 2_δ) ]
```

Our new SOW can then be represented as

$\boxed{6}$ [3/δ].

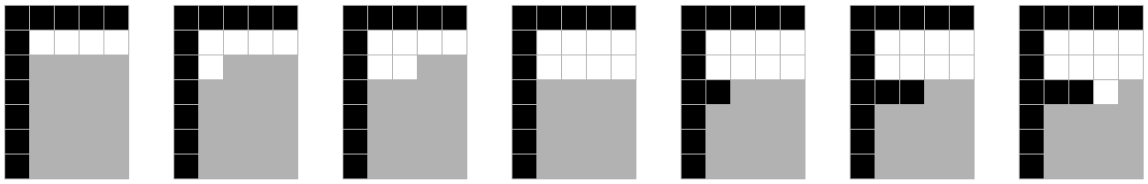

Figure 12: States of work.

Step 13. The same sort of thing applies again.

$\boxed{6}$ [γ →TopRow()]

Since we will be repeatedly using the same construction, I will introduce a template for the resulting structures:

G(x) ::= $\boxed{6}$ [γ →x]

(Perhaps at an implementational level, this corresponds to building a fresh chunk structure (one where the γ position is actually occupied by x) rather than continuing to define new objects by reference chunk structure $\boxed{6}$, whose particular details are not of continuing interest.)

The SOW then stands as

```
G(TopRow()).
```

Step 14. The next step is easy:

```
G(UpperLeft-LShape(width: full, height: 2))
```

Step 15. Here we introduce a new type of shape-representing chunk, which I hope is self-explanatory.[53]

```
G(UpperLeft-UtahShape(width: full, height: 2, a: 2))
```

Step 16. Another simple one:

```
G(Top-Rect(height: 2)₇)
```

Step 17. At this point our picture starts to look like an "F", but seeing as how the whole point of the exercise is to learn what an "F" looks like, we will assume there is no pre-existing "F shape" concept.

```
SOW
  black: [[3], Pixel(location: <ImmedRightOf([3].vertical); y=3>₈)₉]        (23)
  white: [7]
```

Several things to note here:

[53]Utah is a state in the U.S. with a similar shape.

- We have used a new notation for the location of the lone pixel:

  ```
  <ImmedRightOf(3.vertical); y=3>
  ```

  This is meant to suggest that the location is constrained by two pieces of information. First, it is somewhere immediately right of the vertical part of $\boxed{3}$. Second, its y coordinate is 3.

- We need to assume that references to tags ($\boxed{7}$ in this case) refer to the entities described by the referenced chunks—not to the chunks themselves. We need to do this because the chunk structure `Top-Rect(height: 2)` only refers to the correct rectangle by virtue of the containing "`in`" context (see (22)), and that context does not appear in (23).

- Instead of reusing $\boxed{7}$, we also could have reframed the white rectangle in the context of the current region, for example as

  ```
  Rect(upper-left: 3.innercorner, right: image-edge, bottom: Above(9))
  ```

Step 18. For variety, this time we will show another possible way of stating a SOW, this time using an "`also-scanned`" field (so that the full set of pixels scanned so far is the union of the black ones and the also-scanned). This is more compact than explicitly stating which pixels were white.

```
SOW
  black: [3, HBar(left: 8, width: 2)10]
  also-scanned: Top-Rect(bottom: Above(9))
```

Step 19. At this point we can most compactly represent the situation by thinking of an ongoing sweep through all the pixels, row by row, top to bottom and left to right. `PixelsUpTo(last: P)` will represent the pixels visited in such a sweep, up to the position P.

```
SOW
  black: 11
  also-scanned: PixelsUpTo(last: RightOf(10))
```

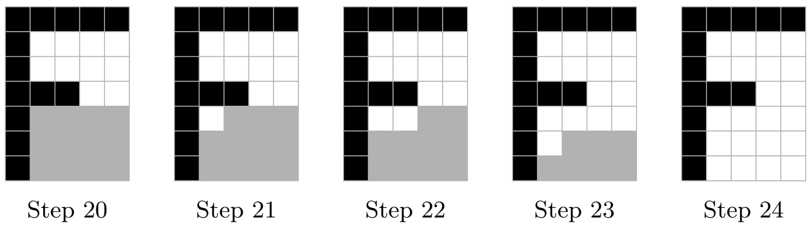

Figure 13: States of work.

Step 20. At this point let us define

```
F(x) ::= SOW
           black: 11
           also-scanned: x.
```

Then the present SOW becomes

```
F(Top-Rect(bottom: SameAs(9))).
```

Step 21. We now continue along similar lines.

```
F(PixelsUpTo(last: <ImmedRightOf(3.vertical); Below(9)>)
```

Step 22.

```
F(PixelsUpTo(last: <x=2, y=Below($\boxed{9}$)>))
```

Step 23.

```
F(PixelsUpTo(last: <x=1, y=5>))
```

Step 24. Finally we have explored the full image and can just specify the black pixels:

```
SOW
  black: $\boxed{11}$
  scanned: all
```

At this point we can report what the image consists of.

We will leave the story of SOW representations here, but before doing so let me reiterate the point of this exercise: having a concrete story about states the mind goes through while working on this task gives us a basis for further modeling. This includes lower–level modeling of how the foregoing sequence of representations might be generated (discussed a bit in §11.6), extensions to other cases and other tasks, and investigations into the knowledge needed to carry out these tasks and its acquisition.

11.5 Formal model: exploring the first couple pixels

So far we have sketched out the gist of what the AS might do to analyze images and gone into some detail regarding possible representations it might build. But we have not been very precise about the operational details of how its mind performs the proposed behaviors. That is what we will discuss in this section and the next. In this section, we will detail computational mechanisms that could be involved in scanning the first couple pixels. In the next, we will look at how the account updates as new items are found. The fraction of the task explored here is necessarily small, because there is a great deal to be said about the computational process behind even small snippets of thought.

Let's now dive in and consider the first moments of looking at an image. Assume the AS probes the upper left pixel and then the one to its right. Our representation of the computation that occurs here takes the form of a TLOG; consult §9.3 for an introduction to this style of representation. The full TLOG is shown in Figure 14; we will discuss it line by line.

11.5.1 The first pixel

We begin with a GNSF `r` for the overall goal. This then triggers an iteration node (`i1`) for the first pixel and a GNSF (`p1`) for the action of probing that pixel. These two are created by rules 1 and 2 respectively.

Rule 1. If one is starting work on a `ProcessImage` GNSF `g`, create

```
IT ^g
  first        // marks the iteration as being the first
  u = ULP
```

Rule 2. If one is starting work on an iteration `i` in the context

```
g GNSF       ProcessImage
i IT    ^g   u = k
```

then create

```
  GNSF  ^i   Probe(k)
```

```
Line Label Type  Parent Details

   1  r     GNSF   ^     ProcessImage
   2  i1    IT     ^r     u = ULP
   3  p1    GNSF   ^i1     Probe(ULP)
   4        GNSF   ^p1      Say(:probe)
   5  q1    GNSF   ^p1      GetAsCoordinate(ULP)
   6  q1        ::           done v
   7  p1        ::          m = v
   8                        ...
   9  p1        ::          done 0
  10  i1        ::         black // flag indicating current pixel is black
  11        DEC    ^i1     decision to go right next       // subconscious
  12  pr    GNSF   ^i1     PlanToGoRightNext     // triggered by last line
  13                        FormRightOf(into=i1\next)       // "Right of"
  14                         done c
  16                        FormIota(i1, \u, into=c)       // "that"
  17                         done d
  19  i2    IT     ^r     u = c
  21                       Probe(c)
  22                        ...
  23                        GetAsCoordinate(c)
  24                        EvalRightOf(c)
  25                         Get(c, :arg)
  26                          done d
  27                         GetAsCoordinate(d)
  28                         FollowIota(d)
  30                          done ULP
  31                         ...
  32                         done f
  33                        m = f
  34                        ...
  35                        done 0
  36                       black
```

Figure 14: TLOG for scanning the first couple pixels. Indentation has been added to show depth of the call stack. Label, type, and parent fields are suppressed for later lines since you probably get the idea.

Each iteration has a variable `u` which identifies the pixel to be looked at in that iteration. In the first iteration, Rule 1 calls for `u` to be `ULP`, which is a tag for a "permanent" mental object representing the upper left pixel of an image. `ULP` would be associated in LTM with the various ways in which that pixel can be described, such as "pixel $(0, 0)$" or "the upper left pixel." (One can view `ULP` as something like a lexical entry.) The agent then needs to perform the `Probe(ULP)` action. To define the behavior that occurs here, we will simply provide "source code" for `Probe`, which would be something like this:[54]

```
Probe(where):
  say(:probe)
  m = GetAsCoordinate(where)
  say(get(m, :x))                                        (24)
  say(get(m, :y))
  return read()
```

We are assuming that to probe a pixel, the AS must send a message of the form "`probe` x y". Our `Probe` function does just that, but note that it may need to convert its argument to a coordinate pair, which is achieved by the call to `GetAsCoordinate`. In the case at hand, `GetAsCoordinate` would look up LTM records associated with `ULP`, and choose the one which is a coordinate pair. This record, notated as v, would just contain the numbers 0 and 0. One the coordinate pair is obtained, the individual coordinates are extracted and emitted, and then the color of the pixel can be read with the `read()` command (black would be 0 and white 1).

By the way, I put `say(:probe)` first in (24) because it's easy to imagine a human sitting at a computer and typing "probe" before having actually decided what to probe yet. That would be consistent with our code if we assume `where` argument might be evaluated lazily.[55]

Where are arguments stored? One thing to point out is that just because in our TLOG we pass an argument to a function does not necessarily mean the mind needs to store that argument in working memory. For example, observe that `i1`, `p1`, and `q1` all have `ULP` as an argument. This does not mean we need space in working memory for three copies of the tag `ULP`—in fact, we don't need to store it at all, because its value is implicit in the call stack and the underlying code.

11.5.2 The second pixel

After the first pixel is probed, I think there are a variety of plausible hypotheses about how things could proceed (and of course, the mind might do different things on different occasions). Let's discuss these possibilities. We'll assume we go to the next pixel right, so the issues center around how and when that decision is made.

First, the decision for where to go next might be deferred (e.g. until we have already typed the next "probe"). We will assume for the moment that this is not the case.

If it is not deferred, there is still the question of whether it becomes *conscious* before we get to the next iteration.[56] Suppose for now that it does (we will come back to the alternative soon). The conscious manifestation could take several forms. One might see the pixel on the VSSP.[57] One might say to oneself any of "$(1, 0)$", "right of $(0, 0)$", "right of that [meaning 'whatever we just did']", "right of $(0, 0)$, which would be $(1, 0)$", and so on. (We might or might not say a more complete phrase like "let's go right next" or "let's go here" or "let's go to $(1, 0)$.") This conscious representation is then presumably consulted when we do the next probe. I will suppose it gets bound to a variable `next` belonging to `i1` for use in the next iteration. This will be notated as `i1\next`.

Out of all these options, we'll somewhat arbitrarily decide to explore a case where the AS *does* generate a consciously experienced memory structure representing the intended next pixel. We'll assume a hypothetical situation where it generates in inner-speech a phrase along the lines of "right of there" to represent its intention for what to do next, and the conscious memory structure it generates corresponds to that phrase.

[54]Words prefixed with `:` are literal symbols rather than variables.

[55]An explanation of lazy evaluation can be found in Abelson, Sussman, and Sussman (1996).

[56]How do we know a conscious representation of the intended next pixel is possible at all? Because we can narrate our decisions as we make them.

[57]Here I am slipping between talking about a hypothetical HLAI and about a human mind, but since we want to model the former on the latter, this is not really a problem.

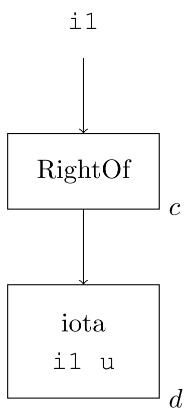

Figure 15: Chunk structure for storing a representation of what pixel the AS intends to look at next.

(The decision would occur at line 11, leading to the GNSF `pr` which builds the conscious structure and stores it in `i1\next`.[58] Note the decision itself is assumed to be subconscious.) What might that structure look like?

My proposal is that it looks something like Figure 15. The basic form is a "RightOf" node with one argument. The argument (the object labeled d) may be thought of as a definite description along the lines of "the value of `u` in iteration `i1`."

Let me explain the motivation for doing things this way. A moment ago I rendered the argument as "there," which of course can't be taken literally since we need to represent where specifically. So the question is what the agent had in mind when using that word (or if not that word, whatever occupies that spot in the inner speech). "There," assuming it's not used deictically, must be getting used either anaphorically (in which case there should be an originating referring expression) or as a sort of "shorthand" for some referring expression which we *could* construct but haven't bothered to. Either way, some sort of (potential or actual) referring expression is involved, so let's postulate that our representation ought to be based off that referring expression. As usual, multiple choices are plausible, but let's assume that the language user is using this mentalese "there" as a shorthand for a definite description along the lines of "the pixel looked at in this turn." This definite description is represented by the object labeled d. It holds a pointer to the iteration in question (`i1`[59]) and the symbol `u`, to indicate that the intention is to refer to the entity that the `u` variable of the iteration holds.

In our TLOG, the above proposal is implemented within the GNSF `pr`, which creates boxes c and d and links them together. At line 19 we get to our next iteration, where the chunk structure just created must be used. Rules for constructing new iteration nodes are shown below.

```
after iteration j in ProcessImage where j\next is not bound, build
  IT                                                                  (25)
    u = pick()     // evaluation of pick() might be deferred or not

after iteration j in ProcessImage where j\next is bound, build
  IT                                                                  (26)
    u = j\next
```

Rule (25) says that if no pixel is already planned for this iteration, then we can pick it. Rule (26), where a plan *does* already exist, "copies"[60] that intended pixel to the new iteration. This is the one that applies in our case, and it causes `i2\u` to be bound to c, as it should be. Just like before, we soon find ourselves at `GetAsCoordinate(`c`)`. Since we see that c is a "RightOf" structure, we dispatch to `EvalRightOf` (and

[58] There may be a sort of bidirectional link here so that it's also understood that `i1\next` is the role this chunk is playing.

[59] Note that even though `i1` has definite descriptions ("the first iteration," "the current iteration," etc.) we don't insist on using one here.

[60] As discussed earlier, no actual, physical copy needs to be made here.

```
EvalRightOf(q'):
  r = form(:match, q')
  s = form(:PixFromCoord, into=r)
  q = GetAsCoordinate(q')
  link(+(get(q, :x), 1), s, :x)
  link(get(q, :y), s, :y)
  return s
```

Figure 16: "Code" for the `EvalRightOf` function.

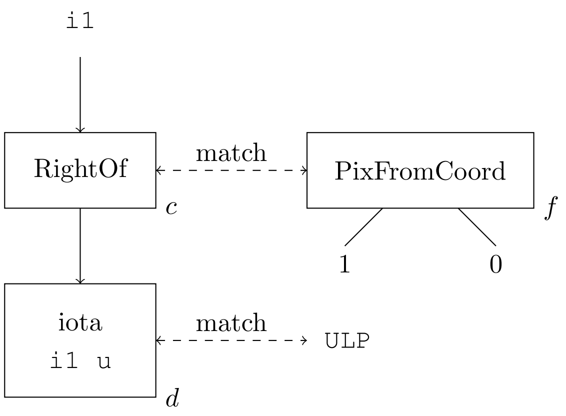

Figure 17: Chunk structure gets annotated once c is "normalized" to a coordinate form.

use tail call elimination). Code for `EvalRightOf` is given in Figure 16. Within that GNSF, d must be unraveled, which is achieved by the `FollowIota` function. A new chunk f is then produced (see code) to represent $(1, 0)$.

Note that when building this $(1, 0)$ chunk, we also build a chunk structure that indicates c matches it. The same holds for d matching `ULP`. This is shown in Figure 17. Future calls to `GetAsCoordinate` would use that record to shortcircuit the work. Once we have $(1, 0)$, we can probe it, and the iteration is complete.

11.5.3 Alternate representational options for the chunk structure

When we finish the first iteration and create the chunk structure RightOf(something) to represent where we intend to move next, there are a number of ways this could be done. I think we can get a reasonable idea of what is possible by considering the things we might say at this point. For example, we might say.

I've just looked at $(0, 0)$. Next let's look right of that. (27)

This usage of "that" seems different from the one we diagrammed; this time, "that" seems to refer pretty directly to $(0, 0)$, and so the chunk structure would presumably look more like Figure 18. There are a number of pesky questions here though. For example: is the pixel chunk g identical to the one permanently stored in LTM associated with `ULP`, or should it perhaps be some "expanded" structure which shares its main content with `ULP`, but also "points to" a record indicating that it came from `i1\u`? The justification for that proposal would be that (27) suggests that $(0, 0)$ has come into play precisely *because* it was what was just looked at. In other words, the fact this pixel is `i1\u` may be its most significant attribute, and thus worthy of being part of whatever structure we use to encode the referent of "that." A version with a copula makes the point even stronger:

The pixel I just looked at was $(1, 0)$. Let's look right of that.

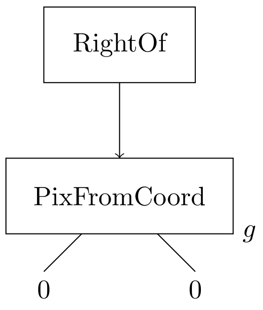

Figure 18: Alternative chunk structure for the intended next pixel.

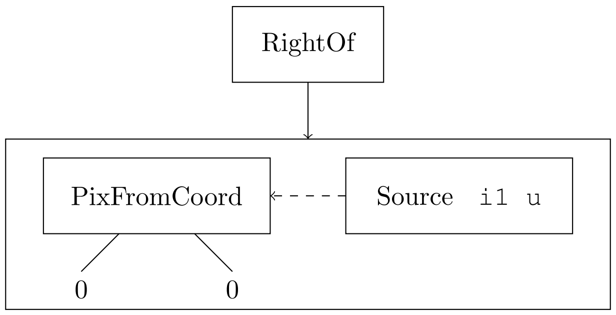

Figure 19: Another alternate chunk structure for the intended next pixel.

Now what does "that" mean? It feels even more ambiguous. Perhaps a reasonable solution here is a structure like Figure 19.[61] (You might point out that even without such a "compound" description for the entity, we can could still infer that g is the same as `i1\u`, since both are associated with `ULP`. But that implicit connectedness feels a bit tenuous.) Another question is whether we should have another type of "aggregate reference" construction for a phrase like "right of that, i.e. $(1, 0)$"—or is this effectively the same as Figure 19?

Yet another interesting case is

I have just looked at my first pixel, which was $(0, 0)$.

Here, "my first pixel" is in a sense nonreferential, as the point of "I have just looked at my first pixel" is understood without needing to resolve the phrase "my first pixel". Nonetheless, "my first pixel" has the *possibility* of being resolved, and so one might want to include both descriptions ("my first pixel" and (0,0)) in one's chunk-based representation of "it" when one says "let's look right of it."

Yet another approach to these problems, mentioned here just for completeness, is that we could have some sort of single node representing `ULP` and everything we know about it.

We will not try to resolve all these questions here; likely more data is needed to do so in a principled way.

11.5.4 Alternative representational options for the intended next pixel

In §11.5.2 we made the assumption that a conscious decision about the next pixel was made before the conclusion of the iteration for the first pixel, but we mentioned that perhaps our minds might also make such a decision *subconsciously*. How would that look?

In this case, we might imagine that `i1\next` gets bound to an object representing a delayed evaluation.[62] We can notate this as (28).

[61] Also worth noting is that the issue of how you describe an entity (whether "$(0, 0)$" or "the pixel we just looked at") is orthogonal to the issue of the significance of the entity. Just knowing that we are now looking at the entity right of the last one doesn't tell us that it's *because* it's right of the last one that we're looking at it.

[62] Again consult Abelson, Sussman, and Sussman (1996), where such objects are called "thunks."

```
delay(compute-right-of(i1\u))
```
(28)

This is copied over to `i2\u` and later evaluated to produce an actual chunk. This evaluation would be triggered in `GetAsCoordinate`, shown below, by the line `force(z)`.

```
GetAsCoordinate(z)
  k = force(z)
  if InCoordinateForm(k):
    return k

  // other cases not shown
```

(Note that if `z` is not a "delay" construct, then `force` will return it unchanged.) With this approach, there may be no "RightOf" chunk ever created. Actually, one wonders whether the explicit chunk creation of the $(1, 0)$ might even be skipped, and the mind might learn to—given (28) and a desire to (say) compute the x coordinate, jump to `(+ (get i1\u :x) 1)`. (Though just the act of typing "1 0" might result in creating a chunk.) Such a shortcut would be a case of automatization, an important aspect of learning but one we will leave for future exploration.

By the way, (28) could really be *any* (subconscious) structure representing the idea of "right of `i1\u`" that could later be "forced" when information about the entity in question was called for. And like `ULP` before, it could exist as code, with no memory usage, something like this:

```
given decision to go right in iteration i:
  activate the assertion
    i\next = [right of i\u]
```

11.6 Formal model: account updates

Let's now turn to a different segment of the story: a piece where the AS completes its scan of a VBAR (a contiguous group of pixels in the same column) and wants to add it to the account it has been tracking. A possible TLOG for that process is shown in Figure 20.

Above the dashed line much is omitted; we're just describing the situation we find ourselves in. More specifically, we are in the process of looking at an image (GNSF `r`). Our account so far, A, consists of an HBAR we have already found (whose details we won't bother specifying), and now we are in the process of exploring a VBAR V which we have discovered.[63] (The GNSF `s` corresponds to this subgoal.) The existence of the VBAR was inferred from noting that $(0, 0)$ and $(0, 1)$ were black. Also note the GNSF `s` tracks the x position it is working at. Although this is inferrable from the chunk V, it seems like it should be kept around so we don't have to re-consult the chunk.

So far the work accomplished within `s` has been to scan down to $(0, 5)$ (all of which pixels were black), and the next thing to do is to check $(0, 6)$. The iteration node `i6` is responsible for doing this check. The `[is bottom]` annotation on that iteration indicates a realization the agent has had that it is at the bottom of the image. At line 7, the agent comes to the conclusion—licensed by the fact that the pixel was black—that V continues at least to the current position (this "coming to a conclusion" needn't be conscious). Because the agent needs to "go somewhere" with this piece of information, I have notated it as a GNSF, which means that it sits at the top of the goal stack and will now be dealt with.

In particular, because we were looking at the bottom of the image, `s` can be updated with the information that the VBAR finishes at the bottom of the image, and having found the extents of the VBAR it can now proceed to its second job, which is to update the existing account A with this new information. To explain what happens here, we will show the code rather than using the TLOG; this code is shown in the same figure.

Lines 16–19 simply show the context in which this code is to run. At line 21, we create a new chunk to represent the updated account—this call specifies the spot where the new chunk is to be stored (namely within `lai\acc`).[64] We then copy the contents of the old account to the new one and add chunk structure

[63] We will not make any commitments as to what that structure looks like.

[64] The `&` notation, borrowed from the C programming language, indicates that we are passing to `CreateAccountChunk` the location of the variable `lai\acc`, not its value.

```
V = <vbar induced by detection of black pixels @ (0,0) and (0,1)>
A = Account
      [ <hbar> ]

Line Label Type Parent Details

  1  r            ^    LookAtImage       acc = A
  2  s            ^r    ScanVBar         x = 0, vb = V
  3  -------------------------------------------------------------
  4  i6    IT     ^s     y=6             [is bottom]
  5  t            ^i6     Probe(0, 6)
  6  t        ::           done 0
  7  u            ^i6     [exists to present]
 10  s        ::         [finishes at bottom]; state = done scanning
 11               ^s     UpdateAccount()
 12  ...
 13  s        ::         done
 14  r        ::        state = pick next action
 15
 16  within
 17   LookAtImage lai(acc = a)
 18    ScanVBar s
 19     UpdateAccount():
 20
 21       a2 = CreateAccountChunk(into=&lai\acc)
 22       for item in a:
 23         Add(a2, item)
 24       v = CreateVBarChunk(into=a2)
 25       link(v, :top, describe(s.vb, :top))
 26       if s.[finishes at bottom]:
 27         link(v, :bottom, :bottom-of-image)
 28       // other case not shown
 29       complete(a2)
```

Figure 20: TLOG and code for `UpdateAccount`.

for the new VBAR. Finally, at line 29, we mark a2 as "complete"; the motivation for this additional step is that if the agent gets interrupted mid-thought, we don't want it to come away with the impression that its account is actually some *subset* of the true account. To make this scheme work, we assume that old values of lai\acc would be kept around (at least for some time), and if the latest is not marked complete, the previous one can be used.

At the end of this process, we end up with a chunk structure along the lines of the following sitting in r\acc:

```
Account
  [ <hbar>,
    VBar(top: (0, 0), bottom: bottom-of-image) ]
```

This concludes the update of the account, and the agent can now move on to other things within the r GNSF.

12 Beyond EFHILT: The case of "A"

The EFHILT letters (at least under our simplifying assumptions) admit relatively simple descriptions. In this section we will consider how the AS might work on a slightly more complicated letter, namely "A". We'll consider (1) the development of a new concept ("diagonal bar") in terms of which the "A" shape will be expressed, (2) what a good representation for the "A" template might look like, and (3) the evolution of the template when a new example is encountered that doesn't quite match the existing understanding.

As before, the AS will analyze letters by building accounts of them. These accounts progress into characterizations of what an "A" consists of in general. There are actually at least a few "styles" in which "A"s are commonly drawn in small bitmap fonts; we will just look at one for now, exemplified by Figure 21.

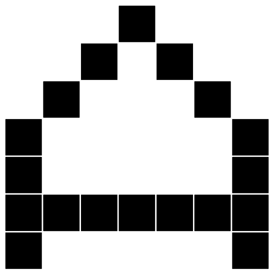

Figure 21: An "A".

12.1 Diagonals

One of the concepts we'll want for analyzing shapes like that of Figure 21 is that of a "diagonal segment"—in this figure, there are two diagonal segments at the top of the image. Although we could just teach the AS about diagonals ourselves, let us instead ponder how it might invent them; the AS will need to do such conceptual invention sooner or later, and this is a good chance to see how it could work.

First we assume that the AS notices an "interesting" group of pixels, namely the ones in red in Figure 22. What makes this particular group interesting? Primarily the fact that it can be generated by starting at one of its pixels and repeatedly moving in the same direction (northeast or southwest). Size is also a factor; the red group is more interesting than the blue one. Another consideration which increases the salience of this particular group of pixels (and therefore would probably make them more likely to be even *considered* for grouphood) is the fact that they are part of a "chain" (a sequence of adjacent pixels).[65,66] We will not at this point dig into how exactly interesting groups come to be found, we simply assume that the low-level details of how the AS explores the image are such that interesting groups tend to get noticed (for example, it would be likely to traverse chains).

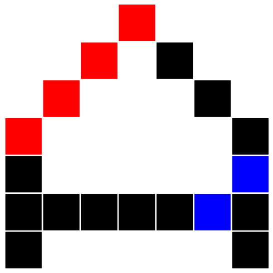

Figure 22: An interesting group of pixels and a less interesting one.

Once the AS finds this interesting group of pixels, and perhaps another similar group of pixels on the right side of the figure, it makes sense to create a new concept—let us call it DBAR—to represent what it is that these groups of pixels have in common (and what makes them interesting). At this point, mind you, the AS needn't assign a name to the concept. A name is an optional ornament (albeit one that a human would probably create sooner or later if she found herself making substantial use of the concept).

It's also important to realize that the concept might develop in the agent's mind (and even be named and articulated verbally) without ever getting a precise definition. For example, the AS may not have considered questions like these: how many pixels need to be present to for something to be a DBAR? Is one enough? Must a DBAR extend as far as possible within an image or is it ok to call three out of ten diagonal pixels its own DBAR? Such decisions about usage needn't be made, at least not yet, because the AS has no need for them; the idea of a DBAR is functioning mainly as a means to generate a compressed representation of the shape, not to draw some sharp distinction between DBARs and non-DBARs. In §13 we will discuss these various issues in more depth.

12.2 Account representation

Now that we have the DBAR concept available, let's consider the process whereby the AS builds an account of an "A", starting with the same example seen above. (Recall that that is its first move when faced with a new letter to study.) Although this process will be similar to what we proposed for the EFHILT task, this time the resulting account is more complicated, with many more constraints, and we must give some thought both to what information deserves to be included in the account and to how it could plausibly be represented (one factor here is that we want to keep the memory burden low). We will focus just on the *final* representation of the account, rather than going through all the intermediate stages as was done in §11.4.3. This also means we will not need invoke some of the sophisticated representational tricks used in §11.4.3 to keep the state of work representation compact.

First, let us arbitrarily decide on what account the AS will produce. I'll assume that it first finds the 4-pixel DBAR in the upper left, then the symmetric 4-pixel DBAR in the upper right, and then, in an effort to account for the leftovers, analyzes them as two 3-pixel VBARs and a 5-pixel HBAR. So now the question is, as the AS scans the image and locates these various parts, what sort of representation of them does it build up in memory?

We shall assume we want to represent where each item is, which for now will mean supplying enough

[65]More technically, what we mean by a chain is a sequence of adjacent pixels $x_1, x_2, \ldots, x_n$ where each x_i $(2 \le i \le n-1)$ has only x_{i+1} and x_{i-1} as neighbors. But note the AS needn't have such a definition (or even the "chain" concept); the fact that it follows chains could simply emerge from a desire to move adjacently as it explores the figure.

[66]Further along in its development, we might also want the AS to realize that being part of a chain (or some similar condition) is important for a diagonal bar deserving to be treated as a unit; otherwise the diagonal of a solid square could be analyzed as one, which would be unnatural.

```
a: dbar+    x1=arbitrary       y1=im.top      (3,0) - (0,3)
            x2=im.left
b: dbar-    p1=a.top           x2=im.right    (3,0) - (6,3)
c: vbar     p1=[S from a.bot]  y2=im.bot      (0,4) - (0,6)
d: vbar     p1=[S from b.bot]  y2=im.bot      (6,4) - (6,6)
e: hbar     p1=[E from c]      p2=[W from d]  (1,5) - (5,5)
            y=arbitrary

ancillary: a.bot = b.bot
```

Figure 23: A sketch of the representation the AS uses for its account of our first "A". See text for explanation.

information to exactly reproduce the image.[67] For each HBAR, VBAR, or DBAR, this means we need to specify three parameters. One important question is whether the relevant information should be specified in absolute terms (i.e. with pixel coordinates) or in relative terms, by saying for instance that the first DBAR begins at the top of the image. We will assume that (when available) relative information is always stored (in a way that can be reliably retrieved), whereas absolute pixel values might be stored, but perhaps in a more tenuous, forgettable fashion.[68] There are two reasons for this preference. First, it is a human-like "memory strategy" (humans would clearly find it easier to remember the relational data than to memorize a large collection of numbers). Second, relational information is more useful when generalizing to other shapes.[69]

Given that we are going to represent relational information, another question is *which* relational description to use for a given position when there is more than one available. For example, the left VBAR of Figure 21 could be described as having its x coordinate at the left of the image, or as having the same x coordinate as the left end of the left DBAR. We will assume that the memory representation of the shape does not store redundant information (as all storage has a cost) and will decide which choice to use in each situation arbitrarily (favoring more compact options where applicable).[70] ,[71] We will assume the agent can later reconstruct alternate descriptions for a given position when desired.

We can now display a representation along the above lines: see Figure 23. (We assume that there is an underlying chunk-based representation for this information, but the information is simpler to present in the tabular form used here.) The five constituent items of the "A" are labeled `a` through `e`. Fields labeled `x`, `y`, or `p` specify respectively the x, y, or both coordinates of the endpoints of the segments. These coordinates are specified either by something like `im.left` (leftmost x position in the image) or by an expression like `[S from b.bot]`, meaning the pixel immediately south of the bottom of item `b`. (`[E from c]` means any pixel directly east from a pixel of `c`.)

A couple more notes on this representation. First, relational information is not always enough to fully specify the position of each bar, and when it is not, we will have our representation explicitly encode what aspects of the bar are *not* pinned down by the relational data (using the `arbitrary` tag). The reason for doing this is that when the AS traverses its memory structure (for example to report on its findings, or to compare another shape against the present one), we want it to be able to distinguish between a situation where some aspect of a part is truly unspecified and a situation where it was simply forgotten. (Even if the AS is designed not to forget, using a representation that does not *rely* on perfect memory seems like the

[67]Further down the line, "where something is" and reproducibility can become decoupled, because there may be a variety of ways to draw something among which we don't care to distinguish. For example, we might only care that a curve is roughly a concave arc from point A to point B, not caring about which particular pixels it passes through. Or we might not care exactly how large we draw an "F" shape as long as its constituents satisfy the right constraints (this point is moot in the present situation since we have assumed a tight bounding box).

[68]In fact there are other options as well, such as deictic references, but we won't go into that here.

[69]The fact that relational information is more useful is something the AS would presumably have to learn, but for now we won't worry about how.

[70]One of the features of the VSSP is that it provides us the ability to not have to worry about such issues. It lets us immediately make the right inferences, e.g. that if A is at `im.left` and B is south of A, then A is at `im.left` too.

[71]How would the AS get an appropriate set of relational information into storage? Mainly by avoiding adding anything that could already have been computed, and by stopping once the shape is fully determined.

more human-like approach.) As mentioned a moment ago, we will also have some (perhaps more forgettable) absolute information, which (if remembered) will allow exact reconstruction even of items that are not fully specified by their spatial relationships.

Also, given that we will be storing relative/qualitative information, why store absolute information as well, except where the relative information does not fully specify the parameters of a bar? This is needed so that the AS can notice things about the image. The absolute information is needed at least momentarily, during processing, so that the agent can keep track of where it is in the figure and can realize where it is looking relative to parts of the figure already discovered. It is also useful as a way of noticing "nonaccidental features"—for example, the fact that the bottom ends of the two DBARs have the same y coordinate—which may require retention for a longer period of time. (That particular fact might even be interesting enough to include in our account of the image, and we will allow for this in our representation as an "ancillary" constraint.[72]) Note also that we could postulate an external memory aid (i.e. a "virtual notebook") to help with some of the memory burden. This does have the downside that coincidences can't be automatically noticed by the mind's associative recall mechanisms, but they could always be checked for deliberately.

There is admittedly a lot of information here, enough that a human would probably find it taxing to remember it all without some additional chunking being imposed (and without using the VSSP, which is in reality what a person would probably do). However, note that it is not quite as bad as it looks: the AS does *not* need to remember a long list of facts (which is very difficult for humans without substantial rehearsal). It only needs to remember a list of five items, and then the rest of the information can be "hung on hooks" associated with those particular items. The AS knows what information to look for (constraints involving the item) and it has a means of determining when it has all been retrieved, so there is not a danger of forgetting something without realizing one has done so. (And obviously we are free to give the AS a superior memory to humans.)

12.3 Using the account

Now, having built an account, as an initial guess the AS could conjecture that an "A" is a shape that has the same parts in the same arrangement.[73] That is, a shape that satisfies all the "relative" constraints about where items are positioned, without necessarily matching any of the absolute positions. In making such a conjecture, the AS is of course making use of pre-existing knowledge about what tends to make for successful conjectures when it comes to letter shapes; a priori it could have turned out that the absolute positions were important after all. We should of course ask where this knowledge came from, but we will table that for the moment.

Let's look at a second "A" (Figure 24) and how the AS would start analyzing it to see if it matches the conjectured template, and if not, how it might adjust the template.

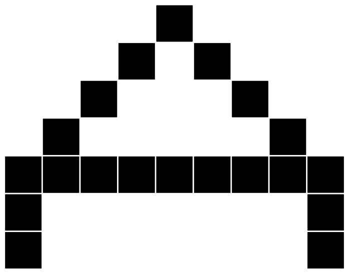

Figure 24: Another "A" image, similar to the last.

Since the first thing mentioned in the account of the first "A" was the DBAR in the upper left, the AS would probably start by hunting for an analogous DBAR here, which is found without much trouble. In the interest of explaining as much data as we can using as little structure as possible (a heuristic we used

[72]We'll avoid introducing redundant constraints though, e.g. not also specifying y-sameness for the `vbar.tops`.

[73]So implicit in our representation (as with any representation) is a mode of generalization.

```
a: dbar+    x1=arbitrary      y1=im.top        x2=im.left
b: dbar-    p1=a.top          x2=im.right
c: vbar     p1=a.bot          y2=im.bot
d: vbar     p1=b.bot          y2=im.bot
e: hbar     p1=[E from c]     p2=[W from d]
            y=arbitrary

ancillary: a.bot = b.bot
```

Figure 25: Template for what constitutes an "A" (or more precisely, one class of "A"s.

implicitly in the first "A"), we include all five pixels as our DBAR. We can then proceed to the other parts we think an "A" has, checking as we go that they are positioned in the way stipulated by our earlier account.

In doing this on our second "A", we hit no trouble until we come to the HBAR (which we either go looking for, since we expect there to be one, or else realize exists because the leftover pixels have that shape). The problem with the HBAR that we find in this new "A" is that it doesn't quite match all the same relationships that the initial HBAR did: its y coordinate is outside the range of the VBARs.

There are two ways out: we can either drop that constraint, or we can re-analyze the VBARs to extend one pixel higher (that reanalysis doesn't mean we *redo* the analysis, only that we alter the results in a way we know to be legitimate). The latter option is preferred because, we now realize, there was actually no good reason why we didn't extend the VBAR as far as possible in the first place. (In fact we can see that if we had analyzed the first "A" in a different order, we would have ended up with the longer VBAR.) Under this new analysis we can see that the constraint on the HBAR will no longer be a problem.

At this point we are likely to re-traverse the whole template to make sure our reanalysis is correctly reflected in it. We may have already realized the need to tweak our template to have the VBARs use the constraint `c.top = a.bot` (and likewise on the other side[74]), but to obtain confidence that the full template is correct, and to make sure we remember the new version, we would probably want to revisit/rehearse the whole thing. (Specifically we would be on the lookout for any references to our VBARs that might have to change, but it turns out there are none.) The result is something like Figure 25. This constitutes a template that will suffice for a variety of "A"s that look similar to Figure 21. (By the way, the mental structure now built up now becomes saved not under "A #1" but under "conjectured pattern for 'A's in general known to work for both A #1 and A #2.")

One more point before we move on. Having done the reanalysis discussed a moment ago, if the AS is forward thinking, it will realize that building accounts by the "leftovers" methodology (where only pixels not accounted for yet are parsed) may not be the best strategy. Instead, it may be productive to extend bars when possible; this is a way of *normalizing* the accounts so that they are less dependent on the particular order of analysis, and it also allows (as we saw here) for more generality. So what we have here is a case of a highly general heuristic being acquired through engagement with a particular problem. I suspect this may in fact be how many heuristics develop.

13 On the flexible nature of concepts

One thing missing from previous discussions is an analysis of the imprecise, soft, fuzzy aspect of concepts. To understand what we mean by this, consider the following assumptions one might naively make about concepts:

- For any given concept, any given entity either *is* or *isn't* an instance of that concept.

[74]In reality, the AS would probably store its analysis in a way that exploited the symmetry, e.g. by remembering that the strutures on the right were analogous to the left, but we will not dive into that here.

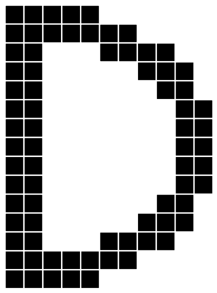

Figure 26: Bitmap image of a "D".

- A concept has a precise definition (using first-order logic, say), and for any concept we use we know what that definition is.
- Knowing that X is an instance of concept Y doesn't tell us anything useful about X except for what can be inferred with certainty from the definition of Y.

In general, none of these things are true of concepts as used by humans, all the more so when the concept is one we've just invented (which is the case with conceptual invention by our AS). Concepts usually have a "prototype" structure where certain features are shared by their most prototypical instances, but even instances which only share some of the features can still be labeled by the concept. For example, the prototypical book is organized into chapters, is reasonably long (perhaps 200 pages or more), has a bunch of rectangular pages printed on both sides and bound with a hard or soft cover, and is one of many identical copies. But if some of those criteria weren't met, we might still call it a book. There are a few contexts where humans use fully precise concepts (most notably mathematics[75]), but these are the exception rather than the rule.

So far we have largely been able to dodge questions about the exact nature of concepts we've used (like HBAR, VBAR, and DBAR), but more refined formalizations of concepts (which will be necessary for HLAI sooner or later) will likely bring us face to face with the sorts of issues just raised. So in this section we will discuss how the question of fuzziness fits into the microscience domain, and what it implies regarding the representation and development of concepts.

13.1 Flexible concepts for analyzing letters

To illustrate a flexible concept in context, consider the image in Figure 26. This is an image that I decided to analyze in order to observe my own thoughts while doing so, so essentially I was playing the role of the AS. I had already looked at many other similar images of letters, and so I had a general sense of what letters in my data set looked like—that is, I was not coming in "naive" as to what letters looked like, a point that will become relevant in a moment.

As I looked at this image, one of the first things I found myself saying was (29).

$$\text{At the left of the image we have a vrect.} \qquad (29)$$

"Vrect" was actually a term I made up on the spot, to denote something along the lines of a "long thin vertical rectangle."[76] What I want to investigate here is the content of this "vrect" concept. If we are to understand how, when, and why observations like (29) are made, we need to say something about what a concept like "vrect" is.[77]

[75]Even in math, "perfectly precise concepts" often have their basis in imprecise ones. For example, words like "space," "partition," or even "number" don't have precise definitions—it's only when one says something like "vector space," "partition of an interval," or "real number" that one identifies an exact meaning.

[76]"Vertical" meaning the height is more than the width.

[77]One complication here is that I intend to use "vrect" to refer to rectangles that we would naturally individuate in the image—so that a black rectangle surrounded by white is fine, but one surrounded by black is not. I don't wish to deal with enforcing this constraint at the moment; instead I'll be concentrating more on the metric properties of vrects: how big do they have to be and so forth.

The first thing to point out is that even though the *name* "vrect" was a spontaneous creation, the underlying concept it named did not seem to be. I felt it already had a "definite existence", as if I was just giving a name to an idea I was already familiar with. (Perhaps I was already using this concept implicitly, or perhaps it met certain naturalness criteria indicating it was worthy of a name.) One can see this in that I would have felt comfortable defining "vrect" (at least to myself) ostensively, by just highlighting one example pixel set and saying "this sort of thing." So the question isn't so much what was on my mind as I introduced the word "vrect" as it is what the internal structure of the concept *named by* "vrect" was, and how it came to be.

Let me first dispense with some proposals that I think fail to do the concept justice. First, one might be inclined to declare that a "vrect" is simply a rectangle whose height is higher than its width, but I don't feel this is at all what I meant. For example "vrect" carries a more specific connotation, one which includes a high aspect ratio; I would only have admitted a 5×4 rectangle as an "atypical" vrect, if at all, and if I had to describe such a shape I would probably choose some other term.

A less naive idea is that perhaps when I applied the term "vrect", I simply meant a "long thin vertical rectangle" (that's the paraphrase I used to explain the idea to you, after all). There are two problems with this. First, what are "long" and "thin" to mean? Such terms are context-dependent. Second and perhaps relatedly, intuitively I don't feel that "long thin vertical rectangle" has the same informational content that "vrect" did. "Long thin vertical rectangle" feels overly generic and therefore divorced from the particular context at hand. That context involves a particular type of data (letter forms on grids with certain characteristic sizes) and indeed a particular set of data. Within that context, the particular sort of "long thin vertical rectangle" we end up using "vrect" to refer to tends to have certain characteristics (for example, having a length of at least 5 pixels[78]). More generally, we might say that "vrect" has not just an intensional aspect (the sort of thing we can capture with "long thin vertical rectangle") but an "extensional" aspect, one which captures "the sort of things that this concept tends to be used for". Attempting to define "vrect" with that paraphrase does not capture the extension at all.

13.2 How can a flexible concept be represented?

So far we have concluded that "mere definitions" aren't enough to fully capture what we mean by a concept (although surely they are at least part of the story), because the concept also means, partially, "whatever it is that the items I've described (or heard described) by that concept had in common." I'll use the term "extensional basis" for all those past instances that helped form our impression of the concept. The main question at this point is how the mind's representation of the concept would "capture" the extensional basis, as clearly it must if there is to be a difference between saying "vrect" and saying "long thin vertical rectangle". However it does it, it seems at best partially accessible to consciousness; we can make determinations that (for example) a height of 3 would be atypically short for a vrect, but we can't "read off" what our minds actually mean by "vrect" in any fully explicit sense.

I think this may be where semantics meets statistical learning, as exemplified by machine learning. Recall that a major type of unsupervised machine learning is clustering. A clustering algorithm invents a bunch of bins into which it groups the items it has seen (in such a way that "similar" items wind up in the same bin), and often implicitly produces a classifier which can sort a new sample into one of the bins it came up with. If we think of our experience working with various shapes in the letter domain as providing a variety of samples to our brain for clustering, then a concept like "vrect" could be viewed as one of the resultant bins that results from such clustering—the one with the long thin vertical rectangles. This bin would capture what the definition does not: the statistical distribution of "vrects."[79] And the fact that "vrect" has a linkage to this (subconscious) bucket would explain the intuitive feeling that "long thin vertical rectangle" (and perhaps *any* paraphrase) does not do the concept justice. Even if, hypothetically, one could find an explicit definition for the concept that exactly matched one's "classifier-driven" intuitive judgments (including judgments about

[78]But note that this "at least 5" was basically a post-hoc judgment about what the concept signifies—it is not as if, prior to making that judgment, I had any particular numerical model consciously in mind for vrects. And even now that I have mentioned the number 5, I am pretty non-committal about it.

[79]Aspects of the distribution that we might want to capture include means, correlations, and maximal and minimal values of numerical parameters (which may be fuzzy, to allow for graded judgments along a typical/atypical/borderline/nonmember axis).

degree of typicality), one would still feel reluctant to declare that that definition was the be-all and end-all of the concept, because one wouldn't *know* that it in fact matched one's classifier.

So that will be our working hypothesis: based on experience with past items, a mind (natural or artificial) forms subconscious "bins" which store the statistical characteristics of a group of similar items. These bins then become part of any concept we later decide to invent.

By the way, although in the case at hand I started using "vrect" without having come up with any explicit definition for it, we can certainly create and use explicit concepts if we wish. For example we could arbitrarily declare that by "vrect" we will mean any rectangle of height at least 4 with a height-to-width ratio of at least 2. Such "definitions by declaration" occur in a number of contexts, for example in law, in philosophy, or when performing statistical analyses, where one might say things like "Among young adults (defined as ages 18-25), accident rates are declining." But your average everyday concept never gets an explicit definition.

Another important point is that even when we are relying on our "inner pattern classifiers" to pin down some concept, we still have a sort of "last word" on when to apply or not apply a concept. For example, suppose that so far we have seen ten vertical rectangles (perhaps unified by something beyond just being rectangles, like the role they play in a letter), each between 7 and 15 pixels tall, and we have started using the word "vrect" to refer to this emerging cluster. Now if we encounter a rectangle that is 4 pixels tall, our inner pattern classifier may tell us that this is atypically short for a vrect, but it is up to us whether to apply the label "vrect" or not (or perhaps call it a "short vrect"). The "boundaries" of the concept (like "7 to 15 pixels") are thus subject to revision as new data comes in, if we feel that there is no principled basis for the old boundary. And because we get to make these decisions consciously, our subconscious buckets are partly influenced by our choices.

Along similar lines, it is up to us whether or not to group the first few instances that ultimately turn into a concept; doing so stems from some sort of observation that they have something in common, and the commonality that exists can then be viewed as the "intensional" component of the concept. For example, if we observe that there have been several long thin vertical rectangles (for some definition of "long" and "thin") and that these "deserve" grouping due to playing a similar role in letters, then "long thin vertical rectangle" becomes the intensional "core" of the concept that grows up around those initial examples.[80] The extensional part of the concept then captures whatever other commonalities that items added to our group (already or in the future) tend to have in common.

In keeping with the dynamic process outlined above, we can further imagine that a concept can have a "lifecycle" during which the scope of items we intend it to cover, and the commonalities we intend to ascribe to them, evolve in response to the purposes for which we are using the concept. At various times a concept may seem firmer and at other times we may be relatively noncommittal about its boundaries or about exactly which commonalities we are trying to get at (perhaps they correlate and we haven't found any reason to tease them apart yet). And of course concepts might split, merge, or die, if a distinction starts to seem relevant or irrelevant or the concept as a whole ceases to help us in our thinking.

13.3 Flexible concepts for an HLAI

To summarize the discussion so far, we have observed that when humans investigate data sets[81], one element of our thinking process is that we start finding clusters of similar objects. We can think of these clusters as concepts, we can give them names (e.g. "vrect"), and they have both an intensional and an extensional aspect. We use such concepts to help characterize new situations, for example a new "D" image.

What does all of this mean for HLAI? Well, I think the main conclusion is that we should give the HLAI the ability to use flexible concepts. The challenge is to operationalize that idea. For instance, what information should we store under "vrect" that would allow that label to be confidently applied to a new group of pixels? That's what we turn to next.

I'll tell one possible story for the origin of the "vrect" concept. Here we assume the concept is invented by the agent, although of course it could have come from humans as well.

[80]Why not also have "playing a role in a letter" as part of the intention? Perhaps partly because of vagueness, and/or because we want to make the concept bottom-up computable.

[81]By "data sets" I mean collections of many things of the same sort, which then need to be classified, taxonomized, and so on.

Suppose we look at a bunch of “I” images. This leads us to observe the following:[82]

> Most of these “I”s have a rectangle in the center of the image with width 1 or 2 and height in the 5–14 range. (30)

This observation expresses a “grouping” of objects that have certain commonalities; our making the statement indicates that this grouping seems at least somewhat useful. Without trying to be too explicit about what distinguishes a “mere grouping” from a “concept”, let us just say that a grouping like the one above has at least the *potential* to be a concept.

When we form a concept out of a grouping like the one above, the concept has several pieces of information associated with it. Most obviously, it has a collection of commonalities possessed by the items that led to the construction of the concept. In this case, the commonalities of the rectangles include:

- All are found near the center of the image.
- All are found in “I” images.
- All have a width of 1 to 2.
- All have a height of 5 to 14.[83]
- Possibly additional observations regarding how the rectangles relate to the rest of the image, for example the fact that there are often horizontal rectangles connected at the top and bottom.

Some commonalities, of course, are better than others (both in forming groups and in forming the resultant concepts). Numerical commonalities are more sensible when they are a single range of values (as opposed to saying that a collection of rectangles have in common that their heights are 6, 9, or 11).[84] And commonalities can often be intermixed to form other commonalities; in this case for example we could say that the aspect ratios are always at least 2.5.

Another relevant piece of information for the new concept is what actual items made up the grouping (the extensional basis). Even though we likely won’t remember all the particulars of the extensional basis, we may well remember certain information such as the rough number of elements, or other statistical measures like standard deviation—even if we don’t articulate these things in a thought like (30). Humans seem to track this sort of data subconsciously, in that (at least in some contexts) we can report a ballpark figure of how many items of a given type we have seen.

Somewhat less obviously, we may have some ideas about *which* commonalities we actually care about—that is, which ones we might use to accept or reject future rectangles as instances of the concept, and which ones are taken to be incidental commonalities of the extensional basis. For example, typically when we create shape concepts we want them to be translation-invariant, and we don’t care about the context in which they are found. So here, it is probably just the metric commonalities (the ones regarding height and width) that become “criterial.” However, the fact that the extensional basis has other commonalities is important, because it buttresses the utility of the concept. For example, we can say “I images tend to have vrects in the center.”

Even among the commonalities we do care about, we might care about them to differing degrees, or be in a state of uncertainty regarding how much we care about each one. (And these things can potentially change over the life of the concept—the concept can actually be seen as a bundle of other concepts it might morph into as those decisions are made, and then the decisions would (ideally) be made so as to arrive at a maximally useful version.) We might for example not care too much about the “14” (i.e. not treat it as an upper bound on vrect heights) because there may be no competing concepts to use for taller rectangles, or because we haven’t seen taller rectangles, so the distiction doesn’t seem useful yet. And of course the “5” could be regarded as a hard or soft limit depending on the situation.

Our idea of the “vrect” concept would be an amalgamation of all the information just discussed: explicitly recognized commonalities that led to the concept’s creation, the statistical characteristics and commonalities of the extensional basis, and judgments about how much various commonalities matter.

[82]Again we are assuming that by “rectangle” we mean one that is prominent in the image, without digging into the details of this.

[83]Let’s assume for now that letters needn’t take up full image.

[84]Also, 5 and 14 presumably aren’t hard and fast limits.

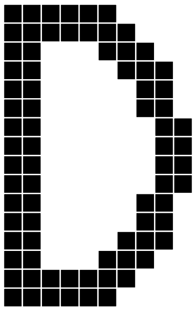

Figure 27: Another "D" image.

13.4 Using a flexible concept

Now that we have it, we can use our "vrect" concept for future rectangles that meet the metric conditions. If a particular rectangle *doesn't* meet the conditions, we can choose not to apply the vrect label, or we might choose to extend the conditions a bit (especially if it's a close call). Or we might defer judgment until we've gathered more information. For example, after looking at "I"s, if we switch to some other letter and find a rectangle that is 3×1, we might want to look at a bunch more of these new letters to see if calling these new rectangles vrects seems warranted. Do they play a useful role in describing the shape of the new letter? As another example of this phenomenon, if we have based our idea of a vrect on "I"s, then the rectangles in question are "clear" on the top and bottom, meaning none of the pixels above or below them are black.[85] The same is true if we then look at the left side of "D"s. But now consider the right side of the "D" in Figure 26, which contains rectangles that meet the metric conditions for vrects but do not have the "clear" property. Should we consider these "vrects"? Obviously it's up to us, but the decision (if and when we make it) will impact the definition of vrect henceforth, as well as what having a vrect will allow us to infer.

Notably, often we don't actually need to make a decision about whether a particular object is an instance of a concept or not. This is for two reasons. First, often the matter won't even come up, because what could have been framed as a rectangle might instead have been conceptualized in some other way first. For example, there are two 3×2 rectangles in Figure 27, but it seems quite possible that we might have analyzed these as part of a "stairstep" pattern, rather than thinking of them as rectangles in their own right, and if so then whether to call them vrects is moot. It's not as if we need to recognize all candidate instances of a concept, let alone judge whether the label should apply.

Secondly, even if a certain rectangle *has* come to our attention, we can just decline to apply the vrect label without actually declaring it's not one. Often the problem we face is not so much to decide whether or not some object is an instance of a category so much as to find *some* category that reasonably captures the properties of the object. If we have something that seems intermediate between a cup and a bowl, choosing to call it a cup doesn't mean it's not a bowl. In general, declining to apply a label does not lead to anything catastrophic; it simply means we can't use the label to build a compact representation of what we're looking at or to draw certain inferences that label may have permitted. (And declining to apply a label, or judging the application borderline, can propagate to later determinations: if a rectangle is a borderline vrect, then the larger shape that contains it may be judged a borderline "D".) If for whatever reason something hinges on applying the label or not, then humans are likely to start making definitions by declaration, as discussed a bit earlier.

The vrect concept can still be valuable even if we don't find ourselves explicitly rejecting candidates as being vrects very often, insofar as it provides a label such that when we do apply it, we are capturing useful information in a compressed form.

13.5 Closing remarks

A few final remarks on flexible concepts.

[85] Assume the vrects extend all the way to the top of the image, even if that means they overlap with serifs.

13.5.1 What good are concepts?

We might wonder what good it does us to have a concept like "vrect." I think this has a relatively simple answer: concepts allow us to generalize. The concept is a label that can be applied to yet-unseen shapes that will hopefully capture what's important about them, more so than "rectangle" can. What do I mean by "important"? Well, what's important about a given item is what can be inferred about it. For example, having labeled something as a "vrect" contributes to the inference that one is looking at a "D" shape. These inferences cannot generally be drawn from the particular low-level details present in the image one is looking at, because one probably has no experience with exactly that set of details.

Along similar lines, once we have identified a set of pixels as a vrect, we can talk about things like the "ends" of the vrect, and say that things like "the end of this vrect overlaps the end of that hrect," a statement that could be significant for letter identification but which would be awkward to phrase in terms of rectangles (and even more awkward to phrase in terms of pixel sets).

13.5.2 Concepts don't need to do *all* the work

Just because we've labeled something a vrect doesn't mean we can't also label it "tall" or "in the ballpark of 10 tall" or whatever. (Likely some of this proceeds subconsciously in humans where we just look at something and get an impression stored on our VSSP that contains that sort of information.) So the detail we intuitively feel exists in our internal conceptualization of a particular object doesn't need to be supported fully by a single label we give to it; other labels or even raw numbers can serve to constrain it.

13.5.3 Concepts from other concepts

Suppose we have a pre-existing "vertical" concept which (when applied to a rectangle) means the height is more than the width. (Actually it might go further and say that the concept is most apt when the height is *notably* more than the width.)

If we notice that this concept applies to the grouping of rectangles that led to the vrect concept (in fact it might have helped inspire the grouping in the first place), it can be marked as an "ancestor" concept of vrect. What effects might this have? For one, things we were able to say about vertical rectangles in general (such as saying an hrect is attached at the middle, which is less sensible for rectangles in general) we can now carry over to vrects. Second, in deciding what commonalities should be how criterial, we can be fairly committed to the criterial commonalities that come from the "vertical" concept. (And of course the "vertical" concept inspired the nomenclature "vrect".)

Consider another presumably pre-existing concept, "tall." We can note that the rectangles in our extensional basis for "vrect" are tall (relative to the range of logically possible rectangle heights, or relative to the distribution of heights of shapes encountered so far). "Tall" (or more precisely, an "instantiation" of "tall" that indicates how tall is tall, in this case tentatively identified as about 5 pixels[86]) then becomes another ancestor concept, and we correspondingly see "taller than around 5" as an important part of the "vrect" idea, more so than the observed upper limit of 14, which has no analogous origin in an ancestor concept.

14 Where to from here?

We have now seen a variety of concrete directions we can go with the HLAI research strategy and methodology outlined in §6 and §7. In each case we picked a specific task for a hypothetical HLAI to pursue and, by considering how humans would do that task, worked out plausible representations and cognitive mechanisms. As far as I know, many of the proposals made here are novel, probably owing to the fact that they are aimed at a rarely-pursued combination of three goals: cognitive plausibility, precise formalization, and nontrivial tasks.

I believe progress in HLAI depends on more work of this type (that is, precise and cognitively plausible descriptions of how a HLAI could perform nontrivial tasks). The proposals discussed in this report are just a tiny sample of the tasks and behaviors that could be approached with the same methodology, and we will

[86] "Tall" is sort of a "meta-concept", in that it provides a blueprint for more specific concepts like "tall for a person", "tall for a tree", etc.

need many more before we are able to see the necessary generalizations to work out cognitive architectures adequate for HLAI.

References

Abbott, B. 2010. *Reference.* Oxford: Oxford University Press.

Abelson, H.; Dybvig, R. K.; Haynes, C. T.; Rozas, G. J.; Iv, N. I. A.; Friedman, D. P.; Kohlbecker, E.; Steele, G. L.; Bartley, D. H.; Halstead, R.; Oxley, D.; Sussman, G. J.; Brooks, G.; Hanson, C.; Pitman, K. M.; and Wand, M. 1998. Revised Report on the Algorithmic Language Scheme. *Higher Order Symbol. Comput.* 11(1):7–105.

Abelson, H.; Sussman, G. J.; and Sussman, J. 1996. *Structure and interpretation of computer programs.* Justin Kelly.

Anderson, J. R. 2007. *How can the human mind occur in the physical universe?* New York, NY, USA: Oxford University Press.

Baars, B. J. 1993. *A cognitive theory of consciousness.* Cambridge University Press.

Baddeley, A. D., and Hitch, G. 1974. Working memory. *Psychology of learning and motivation* 8:47–89.

Baddeley, A. 2000. The episodic buffer: a new component of working memory? *Trends in cognitive sciences* 4(11):417–423.

Bergman, R. 1995. *Learning World Models in Environments with Manifest Causal Structure.* Ph.D. Dissertation, Massachusetts Insitute of Technology.

Croft, W., and Cruse, D. A. 2004. *Cognitive linguistics.* Cambridge University Press.

Culicover, P. W., and Jackendoff, R. 2005. *Simpler syntax.* Oxford University Press on Demand.

Drescher, G. L. 1991. *Made-up minds: a constructivist approach to artificial intelligence.* MIT Press.

Fauconnier, G. 1994. *Mental spaces: Aspects of meaning construction in natural language.* Cambridge University Press.

Fodor, J. A. 1983. *The modularity of mind: An essay on faculty psychology.* MIT Press.

Foundalis, H. E. 2006. *PHAEACO: a cognitive architecture inspired by Bongards problems.* Ph.D. Dissertation, Indiana University, Indianapolis, IN, USA.

Ganesalingam, M., and Gowers, W. T. 2013. A fully automatic problem solver with human-style output. *arXiv preprint arXiv:1309.4501.*

Genesereth, M., and Thielscher, M. 2014. General game playing. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(2):1–229.

Guerin, F. 2011. Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review* 26(02):209–236.

Haugeland, J. 1989. *Artificial intelligence: The very idea.* MIT Press.

Hayes, P. J. 1990. The second naive physics manifesto. 46–63.

Hofstadter, D. R.; Mitchell, M.; et al. 1994. The copycat project: A model of mental fluidity and analogy-making. *Advances in connectionist and neural computation theory* 2(31-112):29–30.

Huddleston, R. D., and Pullum, G. K. 2002. *The Cambridge grammar of the English language.* Cambridge University Press.

Hutter, M. 2001. Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. In *European Conference on Machine Learning*, 226–238. Springer.

Kaiser, J. F. 2010. Richard Hamming - You and Your Research. In Tveito, A.; Bruaset, A. M.; and Lysne, O., eds., *Simula Research Laboratory*. Berlin, Heidelberg: Springer Berlin Heidelberg. chapter 6, 37–60.

Kamp, H. 1981. A Theory of Truth and Semantic Representation. *Formal Methods in the Study of Language.*

Mahabal, A. A. 2009. *Seqsee: A Concept-centered Architecture for Sequence Perception.* Ph.D. Dissertation, Indiana University Bloomington.

Markan, S. 2016. System Induction Games and Cognitive Modeling as an AGI Methodology. In *International Conference on Artificial General Intelligence*, 223–233. Springer.

McGraw, Jr., G. E. 1995. *Letter Spirit (Part One): Emergent High-level Perception of Letters Using Fluid Concepts.* Ph.D. Dissertation, Indiana University, Indianapolis, IN, USA.

Meredith, M. J. E. 1986. *Seek-Whence: A model of pattern perception.* Ph.D. Dissertation, Indiana University.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Newell, A., and Simon, H. A. 1972. *Human problem solving.* Prentice-Hall Englewood Cliffs, NJ.

Newell, A., and Simon, H. A. 1976. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM* 19(3):113–126.

Perez-Liebana, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; Lucas, S. M.; Couëtoux, A.; Lee, J.; Lim, C.-U.; and Thompson, T. 2016. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games* 8(3):229–243.

Pinker, S. 2007. *The stuff of thought: Language as a window into human nature.* Penguin.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures (Artificial Intelligence Series).* Psychology Press, 1 edition.

Schmid, H.-J. 2000. *English abstract nouns as conceptual shells: From corpus to cognition*, volume 34. Walter de Gruyter.

Turing, A. M. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Winograd, T. 1972. Understanding natural language. *Cognitive psychology* 3(1):1–191.