Massachusetts Institute of Technology
Artificial Intelligence Laboratory

AI Working Paper 316 October, 1988

How to do Research

At the MIT AI Lab

by

a whole bunch of current, former, and honorary
MIT AI Lab graduate students

David Chapman, Editor

Version 1.3, September, 1988.

Abstract

This document presumptuously purports to explain how to do research. We give heuristics that may be useful in picking up the specific skills needed for research (reading, writing, programming) and for understanding and enjoying the process itself (methodology, topic and advisor selection, and emotional factors).

Contents

1 Introduction

What is this? There's no guaranteed recipe for success at research. This document collects a lot of informal rules-of-thumb advice that may help.

Who's it for? This document is written for new graduate students at the MIT AI Laboratory. However, it may be useful to many others doing research in AI at other institutions. People even in other fields have found parts of it useful.

How do I use it? It's too long to read in one sitting. It's best to browse. Most people have found that it's useful to flip through the whole thing to see what's in it and then to refer back to sections when they are relevant to their current research problems.

The document is divided roughly in halves. The first several sections talk about the concrete *skills* you need: reading, writing, programming, and so on. The later sections talk about the *process* of research: what it's like, how to go at it, how to choose an advisor and topic, and how to handle it emotionally. Most readers have reported that these later sections are in the long run more useful and interesting than the earlier ones.

- Section 2 is about getting grounded in AI by reading. It points at the most important journals and has some tips on *how* to read.
- 3 is about becoming a member of the AI community: getting connected to a network of people who will keep you up to date on what's happening and what you need to read.
- 4 is about learning about fields related to AI. You'll want to have a basic understanding of several of these and probably in-depth understanding of one or two.
- 5 is about keeping a research notebook.
- 6 is about writing papers and theses; about writing and using comments on drafts; and about getting published.
- 7 is about giving research talks.

- 8 is about programming. AI programming may be different from the sorts you're used to.

- 9 is about the most important choice of your graduate career, that of your advisor. Different advisors have different styles; this section gives some heuristics for finding one who will suit you. An advisor is a resource you need to know how to use; this section tells you how.

- 10 is about theses. Your thesis, or theses, will occupy most of your time during most of your graduate student career. The section gives advice on choosing a topic and avoiding wasting time.

- 11 is on research methodology. This section mostly hasn't been written yet.

- 12 is perhaps the most important section: it's about emotional factors in the process of research. It tells how to deal with failure, how to set goals, how to get unstuck, how to avoid insecurity, maintain self-esteem, and have fun in the process.

This document is still in a state of development; we welcome contributions and comments. Some sections are very incomplete. *[Annotations in brackets and italics indicate some of the major incompletions.]* We appreciate contributions; send your ideas and comments to `Zvona@ai.ai.mit.edu`.

2 Reading AI

Many researchers spend more than half their time reading. You can learn a lot more quickly from other people's work than from doing your own. This section talks about reading within AI; section 4 covers reading about other subjects.

The time to start reading is *now*. Once you start seriously working on your thesis you'll have less time, and your reading will have to be more focused on the topic area. During your first two years, you'll mostly be doing class work and getting up to speed on AI in general. For this it suffices to read textbooks and published journal articles. (Later, you may read mostly drafts; see section 3.)

The amount of stuff you need to have read to have a solid grounding in the field may seem intimidating, but since AI is still a small field, you can in a couple years read a substantial fraction of the significant papers that have been published. What's a little tricky is figuring out which ones those are. There are some bibliographies that are useful: for example, the syllabi of the graduate AI courses. The reading lists for the AI qualifying exams at other universities—particularly Stanford—are also useful, and give you a less parochial outlook. If you are interested in a specific subfield, go to a senior grad student in that subfield and ask him what are the ten most important papers and see if he'll lend you copies to Xerox. Recently there have been appearing a lot of good edited collections of papers from a subfield, published particularly by Morgan-Kauffman.

The AI lab has three internal publication series, the Working Papers, Memos, and Technical Reports, in increasing order of formality. They are available on racks in the eighth floor play room. Go back through the last couple years of them and snag copies of any that look remotely interesting. Besides the fact that a lot of them are significant papers, it's politically very important to be current on what people in your lab are doing.

There's a whole bunch of journals about AI, and you could spend all your time reading them. Fortunately, only a few are worth looking at. The principal journal for central-systems stuff is *Artificial Intelligence*, also referred to as "the Journal of Artificial Intelligence", or "AIJ". Most of the really important papers in AI eventually make it into AIJ, so it's worth scanning through back issues every year or so; but a lot of what it prints is really boring. *Computational Intelligence* is a new competitor that's worth checking out. *Cognitive Science* also prints a fair number of significant AI papers. *Machine Learning* is the main source on what it says. *IEEE PAMI* is probably the best established vision journal; two or three interesting papers per issue. The *International Journal of Computer Vision* (IJCV) is new and so far has been interesting. Papers in *Robotics Research* are mostly on dynamics; sometimes it also has a landmark AIish robotics paper. *IEEE Robotics and Automation* has occasional good papers.

It's worth going to your computer science library (MIT's is on the first floor of Tech Square) every year or so and flipping through the last year's worth of AI technical reports from other universities and reading the ones that look interesting.

Reading papers is a skill that takes practice. You can't afford to read in full all the papers that come to you. There are three phases to reading one. The first is to see if there's anything of interest in it at all. AI papers have abstracts, which are supposed to tell you what's in them, but frequently don't; so you have to jump about, reading a bit here or there, to find out what the authors actually did. The table of contents, conclusion section, and introduction are good places to look. If all else fails, you may have to actually flip through the whole thing.

Once you've figured out what in general the paper is about and what the claimed contribution is, you can decide whether or not to go on to the second phase, which is to find the part of the paper that has the good stuff. Most fifteen page papers could profitably be rewritten as one-page papers; you need to look for the page that has the exciting stuff. Often this is hidden somewhere unlikely. What the author finds interesting about his work may not be interesting to you, and vice versa. Finally, you may go back and read the whole paper through if it seems worthwhile.

Read with a question in mind. "How can I use this?" "Does this really do what the author claims?" "What if...?" Understanding what result has been presented is not the same as understanding the paper. Most of the understanding is in figuring out the motivations, the choices the authors made (many of them implicit), whether the assumptions and formalizations are realistic, what directions the work suggests, the problems lying just over the horizon, the patterns of difficulty that keep coming up in the author's research program, the political points the paper may be aimed at, and so forth.

It's a good idea to tie your reading and programming together. If you are interested in an area and read a few papers about it, try implementing toy versions of the programs being described. This gives you a more concrete understanding.

Most AI labs are sadly inbred and insular; people often mostly read and cite work done only at their own school. Other institutions have different ways of thinking about problems, and it is worth reading, taking seriously, and referencing their work, even if you think you know what's wrong with them.

Often someone will hand you a book or paper and exclaim that you should read it because it's (a) the most brilliant thing ever written and/or (b) precisely applicable to your own research. Usually when you actually read it, you will find it not particularly brilliant and only vaguely applicable. This can be perplexing. "Is there something wrong with me? Am I missing something?" The truth, most often, is that reading the book or paper in question has, more or less by chance, made your friend think something useful about your research topic by catalyzing a line of thought that was already forming in their head.

3 Getting connected

After the first year or two, you'll have some idea of what subfield you are going to be working in. At this point—or even earlier—it's important to get plugged into the Secret Paper Passing Network. This informal organization is where all the action in AI really is. Trend-setting work eventually turns into published papers—but not until at least a year after the cool people know all about it. Which means that the cool people have a year's head start on working with new ideas.

How do the cool people find out about a new idea? Maybe they hear about it at a conference; but much more likely, they got it through the Secret Paper Passing Network. Here's how it works. Jo Cool gets a good idea. She throws together a half-assed implementation and it sort of works, so she writes a draft paper about it. She wants to know whether the idea is any good, so she sends copies to ten friends and asks them for comments on it. They think it's cool, so as well as telling Jo what's wrong with it, they lend copies to their friends to Xerox. Their friends lend copies to their friends, and so on. Jo revises it a bunch a few months later and sends it to AAAI. Six months later, it first appears in print in a cut-down five-page version (all that the AAAI proceedings allow). Jo eventually gets around to cleaning up the program and writes a longer revised version (based on the feedback on the AAAI version) and sends it to the AI Journal. AIJ has almost two years turn-around time, what with reviews and revisions and publication delay, so Jo's idea finally appears in a journal form three years after she had it—and almost that long after the cool people first found out about it. So cool people hardly ever learn about their subfield from published journal articles; those come out too late.

You, too, can be a cool people. Here are some heuristics for getting connected:

- There's a bunch of electronic mailing lists that discuss AI subfields like connectionism or vision. Get yourself on the ones that seem interesting.

- Whenever you talk about an idea you've had with someone who knows the field, they are likely not to give an evaluation of your idea, but to say, "Have you read X?" Not a test question, but a suggestion about something to read that will probably be relevant. If you haven't read X, get the full reference from your interlocutor, or better yet, ask to borrow and Xerox his copy.

- When you read a paper that excites you, make five copies and give them to people you think will be interested in it. They'll probably return the favor.

- The lab has a number of on-going informal paper discussion groups on various subfields. These meet every week or two to discuss a paper that everyone has read.

- Some people don't mind if you read their desks. That is, read the papers that they intend to read soon are heaped there and turn over pretty regularly. You can look over them and see if there's anything that looks interesting. Be sure to ask before doing this; some people *do* mind. Try people who seem friendly and connected.

- Similarly, some people don't mind your browsing their filing cabinets. There are people in the lab who are into scholarship and whose cabinets are quite comprehensive. This is often a faster and more reliable way to find papers than using the school library.

- Whenever you write something yourself, distribute copies of a draft of it to people who are likely to be interested. (This has a potential problem: plagiarism is rare in AI, but it does happen. You can put something like "Please do not photocopy or quote" on the front page as a partial prophylactic.) Most people don't read most of the papers they're given, so don't take it personally when only a few of the copies you distribute come back with comments on them. If you go through several drafts—which for a journal article you should—few readers will read more than one of them. Your advisor is expected to be an exception.

- When you finish a paper, send copies to everyone you think might be interested. Don't assume they'll read it in the journal or proceedings spontaneously. Internal publication series (memos and technical reports) are even less likely to be read.

- The more different people you can get connected with, the better. Try to swap papers with people from different research groups, different AI labs, different academic fields. Make yourself the bridge between two groups of interesting people working on related problems who aren't talking to each other and suddenly reams of interesting papers will flow across your desk.

- When a paper cites something that looks interesting, make a note of it. Keep a log of interesting references. Go to the library every once in a while and look the lot of them up. You can intensively work backward through a "reference graph" of citations when you are hot on the trail of an interesting topic. A reference graph is a web of citations: paper A cites papers B and C, B cites C and D, C cites D, and so on. Papers that you notice cited frequently are always worth reading. Reference graphs have weird properties. One is that often there are two groups of people working on the same topic who don't know about each other. You may find yourself close to closure on searching a graph and suddenly find your way into another whole section. This happens when there are different schools or approaches. It's very valuable to understand as many approaches as possible—often more so than understanding one approach in greater depth.

- Hang out. Talk to people. Tell them what you're up to and ask what they're doing. (If you're shy about talking to other students about your ideas, say because you feel you haven't got any, then try talking to them about the really good—or unbelievably foolish—stuff you've been reading. This leads naturally into the topic of what one might do next.) There's an informal lunch group that meets in the seventh floor playroom around noon every day. People tend to work nights in our lab, and so go for dinner in loose groups. Invite yourself along.

- If you interact with outsiders much—giving demos or going to conferences—get a business card. Make it easy to remember your name.

- At some point you'll start going to scientific conferences. When you do, you will discover fact that almost all the papers presented at any conference are boring or silly. (There are interesting reasons for this that aren't relevant here.) Why go to them then? To meet people in the world outside your lab. Outside people can spread the news about your work, invite you to give talks, tell you about the atmosphere and personalities at a site, introduce you to people, help you find a summer job, and so forth. How to meet people? Walk up to someone whose paper you've liked, say "I really liked your paper", and ask a question.

- Get summer jobs away at other labs. This gives you a whole new pool of people to get connected with who probably have a different way of looking at things. One good way to get summer jobs at other labs is to ask senior grad students how. They're likely to have been places that you'd want to go and can probably help you make the right connections.

4 Learning other fields

It used to be the case that you could do AI without knowing anything except AI, and some people still seem to do that. But increasingly, good research requires that you know a lot about several related fields. Computational feasibility by itself doesn't provide enough constraint on what intelligence is about. Other related fields give other forms of constraint, for example experimental data, which you can get from psychology. More importantly, other fields give you new tools for thinking and new ways of looking at what intelligence is about. Another reason for learning other fields is that AI does not have its own standards of research excellence, but has borrowed from other fields. Mathematics takes theorems as progress; engineering asks whether an object works reliably; psychology demands repeatable experiments; philosophy rigorous arguments; and so forth. All these criteria are sometimes applied to work in AI, and adeptness with them is valuable in evaluating other people's work and in deepening and defending your own.

Over the course of the six or so years it takes to get a PhD at MIT, you can get a really solid grounding in one or two non-AI fields, read widely in several more, and have at least some understanding of the lot of them. Here are some ways to learn about a field you don't know much about:

- Take a graduate course. This is solidest, but is often not an efficient way to go about it.
- Read a textbook. Not a bad approach, but textbooks are usually out of date, and generally have a high ratio of words to content.
- Find out what the best journal in the field is, maybe by talking to someone who knows about it. Then skim the last few years worth and follow the reference trees. This is usually the fastest way to get a feel of what is happening, but can give you a somewhat warped view.
- Find out who's most famous in the field and read their books.
- Hang out with grad students in the field.
- Go to talks. You can find announcements for them on departmental bulletin boards.
- Check out departments other than MIT's. MIT will give you a very skewed view of, for example, linguistics or psychology. Compare the Harvard course

catalog. Drop by the graduate office over there, read the bulletin boards, pick up any free literature.

Now for the subjects related to AI you should know about.

- Computer science is the technology we work with. The introductory graduate courses you are required to take will almost certainly not give you an adequate understanding of it, so you'll have to learn a fair amount by reading beyond them. All the areas of computer science—theory, architectures, systems, languages, etc.—are relevant.

- Mathematics is probably the next most important thing to know. It's critical to work in vision and robotics; for central-systems work it usually isn't directly relevant, but it teaches you useful ways of thinking. You need to be able to read theorems, and an ability to prove them will impress most people in the field. Very few people can learn math on their own; you need a gun at your head in the form of a course, and you need to do the problem sets, so being a listener is not enough. Take as much math as you can early, while you still can; other fields are more easily picked up later.

  Computer science is grounded in discrete mathematics: algebra, graph theory, and the like. Logic is very important if you are going to work on reasoning. It's not used that much at MIT, but at Stanford and elsewhere it is the dominant way of thinking about the mind, so you should learn enough of it that you can make and defend an opinion for yourself. One or two graduate courses in the MIT math department is probably enough. For work in perception and robotics, you need continuous as well as discrete math. A solid background in analysis, differential geometry and topology will provide often-needed skills. Some statistics and probability is just generally useful.

- Cognitive psychology mostly shares a worldview with AI, but practitioners have rather different goals and do experiments instead of writing programs. Everyone needs to know something about this stuff. Molly Potter teaches a good graduate intro course at MIT.

- Developmental psychology is vital if you are going to do learning work. It's also more generally useful, in that it gives you some idea about which things should be hard and easy for a human-level intelligence to do. It also suggests models for cognitive architecture. For example, work on child language acquisition puts substantial constraints on linguistic processing theories. Susan Carey teaches a good graduate intro course at MIT.

- "Softer" sorts of psychology like psychoanalysis and social psychology have affected AI less, but have significant potential. They give you very different ways of thinking about what people are. Social "sciences" like sociology and anthropology can serve a similar role; it's useful to have a lot of perspectives. You're on your own for learning this stuff. Unfortunately, it's hard to sort out what's good from bad in these fields without a connection to a competent insider. Check out Harvard: it's easy for MIT students to cross-register for Harvard classes.

- Neuroscience tells us about human computational hardware. With the recent rise of computational neuroscience and connectionism, it's had a lot of influence on AI. MIT's Brain and Behavioral Sciences department offers good courses on vision (Hildreth, Poggio, Richards, Ullman) motor control (Hollerbach, Bizzi) and general neuroscience (9.015, taught by a team of experts).

- Linguistics is vital if you are going to do natural language work. Besides that, it exposes a lot of constraint on cognition in general. Linguistics at MIT is dominated by the Chomsky school. You may or may not find this to your liking. Check out George Lakoff's recent book *Women, Fire, and Dangerous Things* as an example of an alternative research program.

- Engineering, especially electrical engineering, has been taken as a domain by a lot of AI research, especially at MIT. No accident; our lab puts a lot of stock in building programs that clearly *do* something, like analyzing a circuit. Knowing EE is also useful when it comes time to build a custom chip or debug the power supply on your Lisp machine.

- Physics can be a powerful influence for people interested in perception and robotics.

- Philosophy is the hidden framework in which all AI is done. Most work in AI takes implicit philosophical positions without knowing it. It's better to know what your positions are. Learning philosophy also teaches you to make and follow certain sorts of arguments that are used in a lot of AI papers. Philosophy can be divided up along at least two orthogonal axes. Philosophy

> is usually philosophy *of* something; philosophy of mind and language are most relevant to AI. Then there are schools. Very broadly, there are two very different superschools: *analytic* and *Continental* philosophy. Analytic philosophy of mind for the most part shares a world view with most people in AI. Continental philosophy has a very different way of seeing which takes some getting used to. It has been used by Dreyfus to argue that AI is impossible. More recently, a few researchers have seen it as compatible with AI and as providing an alternative approach to the problem. Philosophy at MIT is of the analytical sort, and of a school that has been heavily influenced by Chomsky's work in linguistics.

This all seems like a lot to know about, and it is. There's a trap here: thinking "if only I knew more X, this problem would be easy," for all X. There's always more to know that could be relevant. Eventually you have to sit down and solve the problem.

5 Notebooks

Most scientists keep a research notebook. You should too. You've probably been told this in every science class since fifth grade, but it's true. Different systems work for different people; experiment. You might keep it online or in a spiral notebook or on legal pads. You might want one for the lab and one for home.

Record in your notebook ideas as they come up. Nobody except you is going to read it, so you can be random. Put in speculations, current problems in your work, possible solutions. Work through possible solutions there. Summarize for future reference interesting things you read.

Read back over your notebook periodically. Some people make a monthly summary for easy reference.

What you put in your notebook can often serve as the backbone of a paper. This makes life a lot easier. Conversely, you may find that writing skeletal papers—title, abstract, section headings, fragments of text—is a useful way of documenting what you are up to, even when you have no intention of ever making it into a real paper. (And you may change your mind later.)

You may find useful Vera Johnson-Steiner's book *Notebooks of the Mind*, which, though mostly not literally about notebooks, describes the ways in which creative thought emerges from the accumulation of fragments of ideas.

6 Writing

There's a lot of reasons to write.

- You are required to write one or two theses during your graduate student career: a PhD and maybe an MS, depending on your department.
- Writing a lot more than that gives you practice.
- Academia runs on publish-or-perish. In most fields and schools, this starts in earnest when you become a professor, but most graduate students in our lab publish before they graduate. Publishing and distributing papers is good politics and good publicity.
- Writing down your ideas is the best way to debug them. Usually you will find that what seemed perfectly clear in your head is in fact an incoherent mess on paper.
- If your work is to benefit anyone other than yourself, you must communicate it. This is a basic responsibility of research. If you write well more people will read your work!
- AI is too hard to do by yourself. You need constant feedback from other people. Comments on your papers are one of the most important forms of that.

Anything worth doing is worth doing well.

- Read books about how to write. Strunk and White's *Elements of Style* gives the basic dos and don'ts. Claire Cook's *The MLA's Line By Line* (Houghton Mifflin) is about editing at the sentence level. Jacques Barzun's *Simple and Direct: A Rhetoric for Writers* (Harper and Row, 1985) is about composition.
- When writing a paper, read books that are well-written, thinking in background mode about the syntactic mechanics. You'll find yourself absorbing the author's style.
- Learning to write well requires doing a lot of it, over a period of years, and getting and taking seriously criticism of what you've written. There's no way to get dramatically better at it quickly.

- Writing is sometimes painful, and it can seem a distraction from doing the "real" work. But as you get better at it, it goes faster, and if you approach it as a craft, you can get a lot of enjoyment out of the process for its own sake.

- You will certainly suffer from writer's block at some point. Writer's block has many sources and no sure cure. Perfectionism can lead to writer's block: whatever you start to write seems not good enough. Realize that writing is a debugging process. Write something sloppy first and go back and fix it up. Starting sloppy gets the ideas out and gets you into the flow. If you "can't" write text, write an outline. Make it more and more detailed until it's easy to write the subsubsubsections. If you find it really hard to be sloppy, try turning the contrast knob on your terminal all the way down so you can't see what you are writing. Type whatever comes into your head, even if it seems like garbage. After you've got a lot of text out, turn the knob back up and edit what you've written into something sensible.

  Another mistake is to imagine that the whole thing can be written out in order. Usually you should start with the meat of the paper and write the introduction last, after you know what the paper really says. Another cause of writer's block is unrealistic expectations about how easy writing is. Writing is hard work and takes a long time; don't get frustrated and give up if you find you write only a page a day.

- Perfectionism can also lead to endless repolishing of a perfectly adequate paper. This is a waste of time. (It can also be a way of semideliberately avoiding doing research.) Think of the paper you are writing as one statement in a conversation you are having with other people in the field. In a conversation not everything goes perfectly; few expect that what they say in a single utterance will be the whole story or last word in the interchange.

- Writing letters is good practice. Most technical papers would be improved if the style was more like a letter to a friend. Keeping a diary is also a way to practice writing (and lets you be more stylistically experimental than technical papers). Both practices have other substantial benefits.

- It's a common trap to spend more time hacking the formatter macrology than the content. Avoid this. LaTeX is imperfect, but it has most of the macrology you want. If that's not enough, you can probably borrow code from someone else who has wanted to do the same thing. Most sites (including MIT) maintain a library of locally-written extensions.

- Know what you want to say. This is the hardest and most important factor in writing clearly. If you write something clumsy and can't seem to fix it, probably you aren't sure what you really want to say. Once you know what to say, just say it.

- Make it easy for the reader to find out what you've done. Put the sexy stuff up front, at all levels of organization from paragraph up to the whole paper. Carefully craft the abstract. Be sure it tells what your good idea is. Be sure you yourself know what it is! Then figure out how to say it in a few sentences. Too many abstracts tell what the paper is generally about and promise an idea without saying what it is.

- Don't "sell" what you've done with big words or claims. Your readers are good people; honesty and self-respect suffice. Contrariwise, don't apologize for or cut down your own work.

- Often you'll write a clause or sentence or paragraph that you know is bad, but you won't be able to find a way to fix it. This happens because you've worked yourself into a corner and no local choice can get you out. You have to back out and rewrite the whole passage. This happens less with practice.

- Make sure your paper has an idea in it. If your program solves problem X in 10 ms, tell the reader *why* it's so fast. Don't just explain how your system is built and what it does, also explain why it works and why it's interesting.

- Write for people, not machines. It's not enough that your argument be correct, it has to be easy to follow. Don't rely on the reader to make any but the most obvious deductions. That you explained how the frobnitz worked in a footnote on page seven is not a justification when the reader gets confused by your introducing it without further explanation on page twenty-three. Formal papers are particularly hard to write clearly. Do *not* imitate math texts; their standard of elegance is to say as little as possible, and so to make the reader's job as hard as possible. This is not appropriate for AI.

- After you have written a paper, delete the first paragraph or the first few sentences. You'll probably find that they were content-free generalities, and that a much better introductory sentence can be found at the end of the first paragraph of the beginning of the second.

If you put off writing until you've done all the work, you'll lose most of the benefit. Once you start working on a research project, it's a good idea to get into the habit of writing an informal paper explaining what you are up to and what you've learned every few months. Start with the contents of your research notebook. Take two days to write it—if it takes longer, you are being perfectionistic. This isn't something you are judged on; it's to share with your friends. Write DRAFT—NOT FOR CITATION on the cover. Make a dozen copies and give them to people who are likely to be interested (including your advisor!). This practice has most of the benefits of writing a formal paper (comments, clarity of thought, writing practice, and so forth), but on a smaller scale, and with much less work invested. Often, if your work goes well, these informal papers can be used later as the backbone of a more formal paper, from an AI Lab Working Paper to a journal article.

Once you become part of the Secret Paper Passing Network, you'll find that people give you copies of draft papers that they want comments on. Getting comments on your papers is extremely valuable. You get people to take the time to write comments on yours by writing comments on theirs. So the more people's papers you write comments on, the more favors are owed you when you get around to writing one... good politics. Moreover, learning to critique other people's papers will help your own writing.

Writing useful comments on a paper is an art.

- To write really useful comments, you need to read the paper twice, once to get the ideas, and the second time to mark up the presentation.
- If someone is making the same mistake over and over, don't just mark it over and over. Try to figure out what the pattern is, why the person is doing it, and what they can do about it. Then explain this explicitly at length on the front page and/or in person.
- The author, when incorporating your comments, will follow the line of least resistance, fixing only one word if possible, or if not then one phrase, or if not then one sentence. If some clumsiness in their text means that they have to back up to the paragraph level, or that they have to rethink the central theme of a whole section, or that the overall organization of the paper is wrong, say this in big letters so they can't ignore it.

- Don't write destructive criticism like "garbage" on a paper. This contributes nothing to the author. Take the time to provide constructive suggestions. It's useful to think about how you would react to criticism of your own paper when providing it for others.

There are a variety of sorts of comments. There are comments on presentation and comments on content. Comments on presentation vary in scope. Copy-edits correct typos, punctuation, misspellings, missing words, and so forth. Learn the standard copy-editing symbols. You can also correct grammar, diction, verbosity, and muddied or unclear passages. Usually people who make grammatical mistakes do so consistently, using comma splices for example; take the time to explain the problem explicitly. Next there are organizational comments: ideas out of order at various scales from clauses through sentences and paragraphs to sections and chapters; redundancy; irrelevant content; missing arguments.

Comments on content are harder to characterize. You may suggest extensions to the author's ideas, things to think about, errors, potential problems, expressions of admiration. "You ought to read X because Y" is always a useful comment.

In requesting comments on a paper, you may wish to specify which sorts are most useful. For an early draft, you want mostly comments on content and organization; for a final draft, you want mostly comments on details of presentation. Be sure as a matter of courtesy to to run the paper through a spelling corrector before asking for comments.

You don't have to take all the suggestions you get, but you should take them seriously. Cutting out parts of a paper is particularly painful, but usually improves it. Often if you find yourself resisting a suggestion it is because while it points out a genuine problem with your paper the solution suggested is unattractive. Look for a third alternative.

Getting your papers published counts. This can be easier than it seems. Basically what reviewers for AI publications look for is a paper that (a) has *something* new to say and (b) is not broken in some way. If you look through an IJCAI proceedings, for example, you'll see that standards are surprisingly low. This is exacerbated by the inherent randomness of the reviewing process. So one heuristic for getting published is to keep trying. Here are some more:

- Make sure it is readable. Papers are rejected because they are incomprehensible or ill-organized as often as because they don't have anything to say.

- Circulate drafts for a while before sending it in to the journal. Get and incorporate comments. Resist the temptation to hurry a result into publication; there isn't much competition in AI, and publication delays will outweigh draft-commenting delays anyway.

- Read some back issues of the journal or conference you are submitting to to make sure that the style and content of your paper are appropriate to it.

- Most publications have an "information for authors," a one page summary of what they want. Read it.

- The major conferences choose prize papers on the basis of excellence both of content and presentation from among those accepted. Read them.

- It's often appropriate to send a short, early report on a piece of work to a conference and then a longer, final version to a journal.

- Papers get rejected—don't get dejected.

- The reviewing process differs greatly between journals and conferences. To get quick turn-around time, conferences must review quickly. There is no time for contemplation or for interaction. If you get bounced, you lose. But with a journal, you can often argue with the editor, and with the referee through the editor.

- Referees should be helpful. If you get an obnoxious referee report, you should complain to the program chair or editor. Don't expect much feedback from conference referee reports. But from journals, you can often get excellent suggestions. You don't have to do all of them, but if you don't you should explain why, and realize that it may take further negotiation. In any case, no matter which side of the reviewing process you are on, be polite. You are going to be working with the people whose papers you review as part of a community for the rest of your professional life.

- MIT AI Lab Memos are generally of publishable or near-publishable quality. De facto, Technical Reports are almost always revised versions of theses. Working Papers can be and often are very informal. They are a good way to get a lot of copies made of a paper you'd want to send to a bunch of colleagues anyway. You publish one of these internal documents by getting a form from the Publications Office (just off the eighth floor playroom) and getting two faculty members to sign it.

Like all else in research, paper writing always takes a lot longer than you expect. Papers for publication have a particularly insidious form of this disease, however. After you finally finish a paper, you send it in for publication. Many months later it comes back with comments, and you have to revise it. Then months after that the proofs come back for correction. If you publish several forms of the paper, like a short conference version and a long journal version, this may go through several rounds. The result is that you are still working on a paper years after you thought you were through with it and after the whole topic has become utterly boring. This suggests a heuristic: don't do some piece of research you don't care for passionately on the grounds that it won't be hard to get a publication out of it: the pain will be worse than you expect.

7 Talks

Talks are another form of communication with your colleagues, and most of what we said about writing is true of talking also. An ability to stand in front of an audience and give a talk that doesn't make the audience fall asleep is crucial for success in terms of recognition, respect and eventually a job. Speaking ability is not innate—you can start out graduate life as a terrible public speaker and end up as a sparkling wit so long as you practice, practice, practice, by actually giving talks to groups of people.

Some ways to learn and practice speaking:

- Patrick Winston has a great short paper on how to give talks. He also gives a lecture based on it every January which simultaneously illustrates and describes his heuristics.

- If you feel you are a bad speaker, or if you want to be a good one, take a course on public speaking. An intro acting class is also useful.

- If your advisor's students have regular research meetings, volunteer to talk about your stuff.

- The MIT AI lab has a series of semiformal talks known as the Revolving Seminar. Volunteer to give one if you have something worth turning into an AI memo or a conference paper.

- Learn enough about the Lab's various robotics projects so when your relatives or friends from out of town come you can give them a tour and a little 60 second talk in front of each robot about it. (Your relatives and non-AI friends will usually love this; they won't be so impressed by the intricacies of your TMS.)

- Since revising a talk is generally much easier than revising a paper, some people find that this is a good way to find the right way to express their ideas. (Mike Brady once remarked that all of his best papers started out as talks.)

- Practice the talk in an empty room, preferably the one in which you will deliver it. Studies of context effects in memory suggest that you will remember what you are going to say better if you have practiced in the room you deliver in. Practice runs let you debug the mechanics of a talk: what to say during each slide, moving overlays around smoothly, keeping notes and slides synchronized, estimating the length of the entire talk. The less time you spend fumbling around with your equipment, the more time you have left to communicate.

- Practicing with a mirror or tape or video recorder is another alternative. The lab has all three. They might help debug your voice and body language, too.

- For a relatively formal talk—especially your Oral Exam—do a practice run for half a dozen friends and have them critique it.

- Watch the way other people give talks. There are a lot of talks given by people visiting MIT. Attending such talks is a good way to get a taste of areas you aren't so familiar with, and if the talk turns out to be boring, you can amuse yourself by analyzing what the speaker is doing wrong. (Going to a seminar is also a way to cure the mid-afternoon munchies...)

- Cornering one of your friends and trying to explain your most recent brainstorm to him is a good way both to improve your communication skills, and to debug your ideas.

Some key things to remember in planning and delivering a talk:

- You can only present one "idea" or "theme" in a talk. In a 20 minute or shorter talk the idea must be crystal clear and cannot have complicated associated baggage. In a 30 or 45 minute talk the idea can require some buildup or background. In an hour talk the idea can be presented in context, and some of the uglies can be revealed. Talks should almost never go on for more than an hour (though they often do).

- The people in the audience want to be there; they want to learn what you have to say. They aren't just waiting for an excuse to attack you, and will feel more comfortable if you are relaxed.

- Take at least one minute per overhead. Some people vary in their rate, but a common bug is to think that you can do it faster than that and still be clear. You can't.

- Don't try to cram everything you know into a talk. You need to touch on just the high points of your ideas, leaving out the details.

- AI talks are usually accompanied by overhead transparencies, otherwise known as "slides". They should be kept simple. Use few words and big type. If you can't easily read your slides when you are standing and they are on the floor, they're too small. Draw pictures whenever possible. Don't stand in front of the screen. Don't point at the overhead if it is possible to point directly at the screen. If you must point at the overhead, don't actually touch the transparency since you will make it jerk around.

8 Programming

Not every AI thesis involves code, and there are important people in AI who have never written a significant program, but to a first approximation you have to be able to program to do AI. Not only does most AI work involve writing programs, but learning to program gives you crucial intuitions into what is and isn't computationally feasible, which is the major source of constraint AI contributes to cognitive science.

At MIT, essentially all AI programming is done in Common Lisp. If you don't know it, learn it. Learning a language is not learning to program, however; and AI programming involves some techniques quite different from those used for systems programming or for other applications. You can start by reading Abelson and

Sussman's *Structure and Interpretation of Computer Programs* and doing some of the exercises. That book isn't about AI programming *per se*, but it teaches some of the same techniques. Then read the third edition of Winston and Horn's Lisp book; it's got a lot of neat AI programs in it. Ultimately, though, programming, not reading, is the best way to learn to program.

There is a lot of Lisp programming culture that is mostly learned by apprenticeship. Some people work well writing code together; it depends strongly on the personalities involved. Jump at opportunities to work directly with more experienced programmers. Or see if you can get one of them to critique your code. It's also extremely useful to read other people's code. Ask half a dozen senior grad students if you can get the source code for their programs. They'll probably complain a bit, and make noises about how their coding style is just awful, and the program doesn't really work, and then give you the code anyway. Then read it through carefully. This is time consuming; it can take as long to read and fully understand someone else's code as it would take you to write it yourself, so figure on spending a couple of weeks spread over your first term or two doing this. You'll learn a whole lot of nifty tricks you wouldn't have thought of and that are not in any textbook. You'll also learn how not to write code when you read pages of incomprehensible uncommented gibberish.

All the standard boring things they tell you in software engineering class are true of AI programming too. Comment your code. Use proper data abstraction unless there is a compelling reason not to. Segregate graphics from the rest of your code, so most of what you build is Common Lisp, hence portable. And so on.

Over your first couple years, you should write your own versions of a bunch of standard AI building blocks, such as

- a truth maintenance system,
- a means-ends planner,
- a unification rule system,
- a few interpreters of various flavors,
- an optimizing compiler with flow analysis,
- a frame system with inheritance,
- several search methods,

- an explanation-based learner,

whatever turns you on. You can write stripped-down but functional versions of these in a few days. Extending an existing real version is an equally powerful alternative. It's only when you've written such things that you really understand them, with insight into when they are and aren't useful, what the efficiency issues are, and so forth.

Unlike most other programmers, AI programmers rarely can borrow code from each other. (Vision code is an exception.) This is partly because AI programs rarely really work. (A lot of famous AI programs only worked on the three examples in the author's thesis, though the field is less tolerant of this sloppiness than it once was.) The other reason is that AI programs are usually thrown together in a hurry without concern for maximum generality. Using Foobar's "standard" rule interpreter may be very useful at first, and it will give you insight into what's wrong if it doesn't have quite the functionality you need, or that it's got too much and so is too inefficient. You may be able to modify it, but remember that understanding someone else's code is very time consuming. It's sometimes better to write your own. This is where having done the half-dozen programming projects in the last paragraph becomes real handy. Eventually you get so you can design and implement a custom TMS algorithm (say) in an afternoon. (Then you'll be debugging it on and off for the next six weeks, but that's how it is.) Sometimes making a standard package work can turn into a thesis in itself.

Like papers, programs can be over-polished. Rewriting code till it's perfect, making everything maximally abstract, writing macros and libraries, and playing with operating system internals has sucked many people out their theses and out of the field. (On the other hand, maybe that's what you really wanted to be doing for a living anyway.)

9 Advisors

At MIT there are two kinds of advisors, academic advisors and thesis advisors.

Academic advisors are simple so we'll dispose of them first. Every graduate student is assigned a faculty member as academic advisor, generally in his or her area, though it depends on current advisor loads. The function of the academic advisor is to represent the department to you: to tell you what the official requirements are, to get on your case if you are late satisfying them, and to OK your class schedule. If all goes well, you only have to see your academic advisor in that capacity twice a year on registration day. On the other hand, if you are having difficulties, your academic advisor may be able to act as advocate for you, either in representing you to the department or in providing pointers to sources of assistance.

The thesis advisor is the person who supervises your research. Your choice of thesis advisor is the most important decision you'll make as a graduate student, more important than that of thesis topic area. To a significant extent, AI is learned by apprenticeship. There is a lot of informal knowledge both of technical aspects of the field and of the research process that is not published anywhere.

Many AI faculty members are quite eccentric people. The grad students likewise. The advisor-advisee relationship is necessarily personal, and your personality quirks and your advisor's must fit well enough that you can get work done together.

Different advisors have very different styles. Here are some parameters to consider.

- How much direction do you want? Some advisors will hand you a well-defined thesis-sized problem, explain an approach, and tell you to get to work on it. If you get stuck, they'll tell you how to proceed. Other advisors are hands-off; they may give you no help in choosing a topic at all, but can be extremely useful to bounce ideas off of once you find one. You need to think about whether you work better independently or with structure.

- How much contact do you want? Some advisors will meet with you weekly for a report on your progress. They may suggest papers to read and give you exercises and practice projects to work. Others you may not talk to more than twice a term.

- How much pressure do you want? Some advisors will exert more than others.

- How much emotional support do you want? Some can give more than others.

- How seriously do you want to take your advisor? Most advisors will suggest thesis topics fairly regularly. Some can be depended on to produce suggestions that, if carried out diligently, will almost certainly produce an acceptable, if perhaps not very exciting thesis. Others throw out dozens of off-the-wall ideas, most of which will go nowhere, but one in ten of which, if pursued with vision, can result in ground-breaking work. If you choose such an advisor, you have to act as the filter.

- What kind of research group does the advisor provide? Some professors create an environment in which all their students work together a lot, even if they are not all working on the same project. Many professors get together with their all their students for weekly or biweekly meetings. Will that be useful to you? Are the advisor's students people you get along with? Some students find that they construct important working relationships with students from other research groups instead.

- Do you want to be working on a part of a larger project? Some professors divide up a big system to be built into pieces and assign pieces to individual students. That gives you a group of people that you can talk to about the problem as a whole.

- Do you want cosupervision? Some thesis projects integrate several areas of AI, and you may want to form strong working relationships with two or more professors. Officially, you'll have just one thesis supervisor, but that doesn't have to reflect reality.

- Is the advisor willing to supervise a thesis on a topic outside his main area of research? Whether or not you can work with him or her may be more important to both of you than what you are working on. Robotics faculty at MIT have supervised theses on qualitative physics and cognitive modeling; faculty in reasoning have supervised vision theses. But some faculty members are only willing to supervise theses on their own area of interest. This is often true of junior faculty members who are trying to build tenure cases; your work counts toward that.

- Will the advisor fight the system for you? Some advisors can keep the department and other hostile entities off your back. The system works against certain sorts of students (notably women and eccentrics), so this can be very important.

- Is the advisor willing and able to promote your work at conferences and the like? This is part of his or her job, and can make a big difference for your career.

The range of these parameters varies from school to school. MIT in general gives its students a lot more freedom than most schools can afford to.

Finding a thesis advisor is one of the most important priorities of your first year as a graduate student. You should have one by the end of the first year, or early in the second year at the latest. Here are some heuristics on how to proceed:

- Read the Lab's research summary. It gives a page or so description of what each of the faculty and many of the graduate students are up to.
- Read recent papers of any faculty member whose work seems at all interesting.
- Talk to as many faculty members as you can during your first semester. Try to get a feel for what they are like, what they are interested in, and what their research and supervision styles are like.
- Talk to grad students of prospective advisors and ask what working for him or her is like. Make sure you talk to more than one student who works with a particular advisor as each advisor has a large spectrum of working styles and levels of success in interaction with his or her students. You could be misled either way by a single data point. Talk to his or her first year advisees and his seventh year advisees too.
- Most or all faculty member's research group meetings are open to new grad students, and they are a very good way of getting an idea of what working with them is like.

AI is unusual as a discipline in that much of the useful work is done by graduate students, not people with doctorates, who are often too busy being managers. This has a couple of consequences. One is that the fame of a faculty member, and consequently his tenure case, depends to a significant extent on the success of his students. This means that professors are highly motivated to get good students to work for them, and to provide useful direction and support to them. Another consequence is that, since to a large degree students' thesis directions are shaped by their advisors, the direction and growth of the field as a whole depends a great deal on what advisors graduate students pick.

After you've picked and advisor and decided what you want from him or her, make sure he or she knows. You advisor may hear "I'd like to work with you" as "Please give me a narrowly specified project to do," or "I've got stuff I'd like to do and I want you to sign it when I'm done," or something else. Don't let bad communication get you into a position of wasting a year either spinning your wheels when you wanted close direction or laboring under a topic that isn't the thing you had your heart set on.

Don't be fully dependent on your advisor for advice, wisdom, comments, and connections. Build your own network. You can probably find several people with different things to offer you, whether they're your official advisor or not. It's important to get a variety of people who will regularly review your work, because it's very easy to mislead yourself (and often your advisor as well) into thinking you are making progress when you are not, and so zoom off into outer space. The network can include graduate students and faculty at your own lab at others.

It is possible that you will encounter racist, sexist, heterosexist, or other harrassment in your relationships with other students, faculty members, or, most problematically, your advisor. If you do, get help. MIT's ODSA publishes a brochure called "STOP Harrassment" with advice and resources. The *Computer Science Women's Report*, available from the LCS document room, is also relevant.

Some students in the lab are only nominally supervised by a thesis advisor. This can work out well for people who are independent self-starters. It has the advantage that you have only your own neuroses to deal with, not your advisor's as well. But it's probably not a good idea to go this route until you've completed at least one supervised piece of work, and unless you are sure you can do without an advisor and have a solid support network.

10 The thesis

Your thesis, or theses, will occupy most of your time during most of your career as a graduate student. The bulk of that time will be devoted to research, or even to choosing a topic, rather than to the actual writing.

The Master's thesis is designed as practice for the PhD thesis. PhD-level research is too hard to embark on without preparation. The essential requirement of a Master's thesis is that it literally demonstrate mastery: that you have fully understood the state of the art in your subfield and that you are capable of operating at that level. It is not a requirement that you extend the state of the art, nor that the Master's thesis be publishable. There is a substantial machismo about theses in our lab, however, so that many Master's theses do in fact contribute significantly to the field, and perhaps half are published. This is not necessarily a good thing. Many of us burn out on our Master's work, so that it is notorious that MIT Master's theses are often better than the PhD theses. This defeats the preparatory intent of the Master's. The other factor is that doing research that contributes to the field takes at least two years, and that makes the graduate student career take too damn long. You may not feel in a hurry now, but after you've been around the Lab for seven years you'll want out badly. The mean time from entrance to finishing the Master's is two and a half years. However, the CS department is strongly encouraging students to reduce this period. If a Master's topic turns out to be a blockbuster, it can be split into parts, one for the Master's and one for a PhD.

To get some idea of what constitutes a Master's thesis-sized piece of research, read several recent ones. Keep in mind that the ones that are easy to get at are the ones that were published or made into tech reports because someone thought they extended the state of the art—in other words, because they did more than a Master's thesis needs to. Try also reading some theses that were accepted but not published. All accepted theses can be found in one of the MIT libraries.

PhD theses are required to extend the state of the art. PhD thesis research should be of publishable quality. MIT machismo operates again, so that many PhD theses form the definitive work on a subarea for several years. It is not uncommon for a thesis to define a new subarea, or to state a new problem and solve it. None of this is necessary, however.

In general, it takes about two to three years to do a PhD thesis. Many people take a year or two to recover from the Master's and to find a PhD topic. It's good to use this period to do something different, like being a TA or getting a thorough grounding in a non-AI field or starting a rock and roll band. The actual writing of the PhD thesis generally takes about a year, and an oft-confirmed rule of thumb is that it will drag on for a year after you are utterly sick of it.

Choosing a topic is one of the most difficult and important parts of thesis work.

- A good thesis topic will simultaneously express a personal vision and participate in a conversation with the literature.

- Your topic must be one you are passionate about. Nothing less will keep you going. Your personal vision is your reason for being a scientist, an image or principle or idea or goal you care deeply about. It can take many forms. Maybe you want to build a computer you can talk to. Maybe you want to save the world from stupid uses of computers. Maybe you want to demonstrate the unity of all things. Maybe you want to found colonies in space. A vision is always something big. Your thesis can't achieve your vision, but it can point the way.

- At the same time, science is a conversation. An awful lot of good people have done their best and they're written about it. They've accomplished a great deal and they've completely screwed up. They've had deep insights and they've been unbelievably blind. They've been heros and cowards. And all of this at the same time. Your work will be manageable and comprehensible if it is framed as a conversation with these others. It has to speak to their problems and their questions, even if it's to explain what's wrong with them. A thesis topic that doesn't participate in a conversation with the literature will be too big or too vague, or nobody will be able to understand it.

- The hardest part is figuring out how to cut your problem down to a solvable size while keeping it big enough to be interesting. "Solving AI breadth-first" is a common disease; you'll find you need to continually narrow your topic. Choosing a topic is a gradual process, not a discrete event, and will continue up to the moment you declare the thesis finished. Actually solving the problem is often easy in comparison to figuring out what exactly it is. If your vision is a fifty-year project, what's the logical ten-year subproject, and what's the logical one-year subproject of that? If your vision is a vast structure, what's the component that gets most tellingly to its heart, and what demonstration would get most tellingly to the heart of that component?

- An important parameter is how much risk you can tolerate. Often there is a trade-off between the splashiness of the final product and the risk involved in producing it. This isn't always true, though, because AI has a high ratio of unexplored ideas to researchers.

- An ideal thesis topic has a sort of telescoping organization. It has a central portion you are pretty sure you can finish and that you and your advisor agree will meet the degree requirements. It should have various extensions that are successively riskier and that will make the thesis more exciting if they pan out. Not every topic will fit this criterion, but it's worth trying for.

- Some people find that working on several potential thesis projects at once allows them to finish the one that works out and abandon the ones that fail. This decreases the risk. Others find that the substantial thrashing overhead this engenders is too high, and choose a single topic before starting any work in earnest.

- You may only be interested in a particular subfield, in which case your thesis topic search is narrowed. You may find, though, that there's no faculty member who can supervise a topic in that field whom you are comfortable working with. You may also find that there doesn't seem to be a natural topic to work on in that field, whereas you have good ideas about something else.

- Choosing a Master's topic can be harder than choosing the PhD topic, because it has to be done before you know very much and before you've built much self-confidence.

- One parameter of PhD topic choice is whether to continue working in the same subfield as your Master's, perhaps extending or building on that work, or to switch to another subfield. Staying in the same field simplifies things and probably will take one to two years off the total time to graduation, especially if a PhD-sized topic becomes obvious during the course of the Master's work. But it may leave you "typecast" as someone who does shape-from-shading or circuit analysis; changing fields gives you breadth.

- Topics can be placed in a spectrum from flakey to cut-and-dried. Flakier theses open up new territory, explore previously unresearched phenomena, or suggest heuristic solutions to problems that are known to be very hard or are hard to characterize. Cut-and-dried theses rigorously solve well-characterized problems. Both are valuable; where you situate yourself in this spectrum is a matter of personal style.

- The "further work" sections of papers are good sources of thesis topics.

- Whatever you do, it has to have not been done before. Also, it's not a good idea to work on something that someone else is doing simultaneously. There's enough turf out there that there's no need for competition. On the other hand, it's common to read someone else's paper and panic because it seems to solve your thesis problem. This happens most when you're halfway through the process of making your topic specific and concrete. Typically the resemblance is actually only superficial, so show the paper to some wise person who knows your work and ask them what they think.

- Not all MIT AI Lab theses are about AI; some are hardware or programming language theses. This is OK.

Once you've got a thesis topic, even when it's a bit vague, you should be able to answer the question "what's the thesis of your thesis?" What are you trying to show? You should have one-sentence, one-paragraph, and five-minute answers. If you don't know where you are going, people won't take you seriously, and, worse, you'll end up wandering around in circles.

When doing the work, be able to explain simply how each part of your theory and implementation is in service of the goal.

Make sure once you've selected a topic that you get a clear understanding with your advisor as to what will constitute completion. If you and he have different expectations and don't realize it, you can lose badly. You may want to formulate an explicit end-test, like a set of examples that your theory or program will be able to handle. Do this for yourself anyway, even if your advisor doesn't care. Be willing to change this test if circumstances radically change.

Try a simplified version of the thesis problem first. Work examples. Thoroughly explore some concrete instances before making an abstract theory.

There are a number ways you can waste a lot of time during the thesis. Some activities to avoid (unless they are central to the thesis): language design, user-interface or graphics hacking, inventing new formalisms, overoptimizing code, tool building, bureaucracy. Any work that is not central to your thesis should be minimized.

There is a well-understood phenomenon known as "thesis avoidance," whereby you suddenly find fixing obscure bugs in an obsolete operating system to be utterly fascinating and of paramount importance. This is invariably a semiconscious way of getting out of working on one's thesis. Be aware that's what you are doing. (This document is itself an example of thesis avoidance on the part of its authors.)

11 Research methodology

[This section is weak. Please contribute!]

A research methodology defines what the activity of research is, how to proceed, how to measure progress, and what constitutes success. AI methodology is a jumbled mess. Different methodologies define distinct schools which wage religious wars against each other.

Methods are tools. Use them; don't let them use you. Don't fall for slogans that raise one above the others: "AI research needs to be put on firm foundations;" "Philosophers just talk. AI is about hacking;" "You have to know what's computed before you ask how." To succeed at AI, you have to be good at technical methods and you have to be suspicious of them. For instance, you should be able to prove theorems and you should harbor doubts about whether theorems prove anything.

Most good pieces of AI delicately balance several methodologies. For example, you must walk a fine line between too much theory, possibly irrelevant to any real problem, and voluminous implementation, which can represent an incoherent munging of ad-hoc solutions. You are constantly faced with research decisions that divide along a boundary between "neat" and "scruffy." Should you take the time to formalize this problem to some extent (so that, for example, you can prove its intractability), or should you deal with it in its raw form, which ill-defined but closer to reality? Taking the former approach leads (when successful) to a clear, certain result that will usually be either boring or at least will not Address the Issues; the latter approach runs the risk of turning into a bunch of hacks. Any one piece of work, and any one person, should aim for a judicious balance, formalizing subproblems that seem to cry for it while keeping honest to the Big Picture.

Some work is like science. You look at how people learn arithmetic, how the brain works, how kangaroos hop, and try to figure it out and make a testable theory. Some work is like engineering: you try to build a better problem solver or shape-from algorithm. Some work is like mathematics: you play with formalisms, try to understand their properties, hone them, prove things about them. Some work is example-driven, trying to explain specific phenomena. The best work combines all these and more.

Methodologies are social. Read how other people attacked similar problems, and talk to people about how they proceeded in specific cases.

12 Emotional factors

Research is hard. It is easy to burn out on it. An embarrassingly small fraction of students who start PhD programs in AI finish. AT MIT, almost all those who do not finish drop out voluntarily. Some leave because they can make more money in industry, or for personal reasons; the majority leave out of frustration with their theses. This section tries to explain how that can happen and to give some heuristics that may help. Forewarned is forearmed: mostly it's useful to know that the particular sorts of tragedies, aggravations, depressions and triumphs you go through in research are necessary parts of the process, and are shared with everyone else who does it.

All research involves risk. If your project can't fail, it's development, not research. What's hard is dealing with project failures. It's easy to interpret your project failing as your failing; in fact, it proves that you had the courage to do something difficult.

The few people in the field who seem to consistently succeed, turning out papers year after year, in fact fail as often as anyone else. You'll find that they often have several projects going at once, only a few of which pan out. The projects that do succeed have usually failed repeatedly, and many wrong approaches went into the final success.

As you work through your career, you'll accumulate a lot of failures. But each represents a lot of work you did on various subtasks of the overall project. You'll find that a lot of the ideas you had, ways of thinking, even often bits of code you wrote, turn out to be just what's needed to solve a completely different problem several years later. This effect only becomes obvious after you've piled up quite a stack of failures, so take it on faith as you collect your first few that they will be useful later.

Research always takes much, much longer than it seems it ought to. The rule of thumb is that any given subtask will take three times as long as you expect. (Some add, "...even after taking this rule into account.")

Crucial to success is making your research part of your everyday life. Most breakthroughs occur while you are in the shower or riding the subway or window-shopping in Harvard Square. If you are thinking about your research in background mode all the time, ideas will just pop out. Successful AI people generally are less brilliant than they are persistent. Also very important is "taste," the ability to differentiate between superficially appealing ideas and genuinely important ones.

You'll find that your rate of progress seems to vary wildly. Sometimes you go on a roll and get as much done in a week as you had in the previous three months. That's exhilarating; it's what keeps people in the field. At other times you get stuck and feel like you can't do anything for a long time. This can be hard to cope with. You may feel like you'll never do anything worthwhile again; or, near the beginning, that you don't have what it takes to be a researcher. These feelings are almost certainly wrong; if you were admitted as a student at MIT, you've got what it takes. You need to hang in there, maintaining high tolerance for low results.

You can get a lot more work done by regularly setting short and medium term goals, weekly and monthly for instance. Two ways you can increase the likelihood of meeting them are to record them in your notebook and to tell someone else.

You can make a pact with a friend to trade weekly goals and make a game of trying to meet them. Or tell your advisor.

You'll get completely stuck sometimes. Like writer's block, there's a lot of causes of this and no one solution.

- Setting your sights too high leads to paralysis. Work on a subproblem to get back into the flow.

- You can get into a positive feedback loop in which doubts about your ability to do the work eat away at your enthusiasm so that in fact you can't get anything done. Realize that research ability is a learned skill, not innate genius.

- If you find yourself seriously stuck, with nothing at all happening for a week or more, promise to work *one hour* a day. After a few days of that, you'll probably find yourself back in the flow.

- It's hard to get started working in the morning, easy to keep going once you've started. Leave something easy or fun unfinished in the evening that you can start with in the morning. Start the morning with real work—if you start by reading your mail, you may never get to something more productive.

- Fear of failure can make work hard. If you find yourself inexplicably "unable" to get work done, ask whether you are avoiding putting your ideas to the test. The prospect of discovering that your last several months of work have been for naught may be what's stopping you. There's no way to avoid this; just realize that failure and wasted work are part of the process.

- Read Alan Lakien's book *How to Get Control of Your Time and Your Life*, which is recommended even by people who *hate* self-help books. It has invaluable techniques for getting yourself into productive action.

Most people find that their personal life and their ability to do research interact. For some, work is a refuge when everything else is going to hell. Others find themselves paralyzed at work when life is in turmoil for other reasons. If you find yourself really badly stuck, it can be helpful to see a psychotherapist. An informal survey suggests that roughly half of the students in our lab see one at some point during their graduate careers.

One factor that makes AI harder than most other types of work is that there are no generally accepted standards of progress or of how to evaluate work. In mathematics, if you prove a theorem, you've done something; and if it was one that others have failed to prove, you've done something exciting. AI has borrowed standards from related disciplines and has some of its own; and different practitioners, subfields, and schools put different emphases on different criteria. MIT puts more emphasis on the quality of implementations than most schools do, but there is much variation even within this lab. One consequence of this is that you can't please all the people all the time. Another is that you may often be unsure yourself whether you've made progress, which can make you insecure. It's common to find your estimation of your own work oscillating from "greatest story ever told" to "vacuous, redundant, and incoherent." This is normal. Keep correcting it with feedback from other people.

Several things can help with insecurity about progress. Recognition can help: acceptance of a thesis, papers you publish, and the like. More important, probably, is talking to as many people as you can about your ideas and getting their feedback. For one thing, they'll probably contribute useful ideas, and for another, some of them are bound to like it, which will make you feel good. Since standards of progress are so tricky, it's easy to go down blind alleys if you aren't in constant communication with other researchers. This is especially true when things aren't going well, which is generally the time when you least feel like talking about your work. It's important to get feedback and support at those times.

It's easy not to see the progress you *have* made. "If I can do it, it's trivial. My ideas are all obvious." They may be obvious to you in retrospect, but probably they are not obvious to anyone else. Explaining your work to lots of strangers will help you keep in mind just how hard it is to understand what now seems trivial to you. Write it up.

A recent survey of a group of Noble Laureates in science asked about the issue of self-doubt: had it been clear all along to these scientists that their work was earth-shattering? The unanimous response (out of something like 50 people) was that these people were constantly doubting the value, or correctness, of their work, and they went through periods of feeling that what they were doing was irrelevant, obvious, or wrong. A common and important part of any scientific progress is constant critical evaluation, and is some amount of uncertainty over the value of the work is an inevitable part of the process.

Some researchers find that they work best not on their own but collaborating with others. Although AI is often a pretty individualistic affair, a good fraction of people work together, building systems and coauthoring papers. In at least one case, the Lab has accepted a coauthored thesis. The pitfalls here are credit assignment and competition with your collaborator. Collaborating with someone from outside the lab, on a summer job for example, lessens these problems.

Many people come to the MIT AI Lab having been the brightest person in their university, only to find people here who seem an order of magnitude smarter. This can be a serious blow to self-esteem in your first year or so. But there's an advantage to being surrounded by smart people: you can have someone friendly shoot down all your non-so-brilliant ideas before you could make a fool of yourself publicly. To get a more realistic view of yourself, it is important to get out into the real world where not everyone is brilliant. An outside consulting job is perfect for maintaining balance. First, someone is paying you for your expertise, which tells you that you have some. Second, you discover they really need your help badly, which brings satisfaction of a job well done.

Contrariwise, every student who comes into the Lab has been selected over about 400 other applicants. That makes a lot of us pretty cocky. It's easy to think that *I'm* the one who is going to solve this AI problem for once and for all. There's nothing wrong with this; it takes vision to make any progress in a field this tangled. The potential pitfall is discovering that the problems are all harder than you expected, that research takes longer than you expected, and that you can't do it all by yourself. This leads some of us into a severe crisis of confidence. You have to face the fact that all you can do is contribute your bit to a corner of a subfield, that your thesis is not going to solve the big problems. That may require radical self-reevaluation; often painful, and sometimes requiring a year or so to complete. Doing that is very worthwhile, though; taking yourself less seriously allows you to approach research in a spirit of play.

There's at least two emotional reasons people tolerate the pain of research. One is a drive, a passion for the problems. You do the work because you could not live any other way. Much of the best research is done that way. It has severe burn-out potential, though. The other reason is that good research is fun. It's a pain a lot of the time, but if a problem is right for you, you can approach it as play, enjoying the process. These two ways of being are not incompatible, but a balance must be reached in how seriously to take the work.

In getting a feeling for what research is like, and as inspiration and consolation in times of doubt, it's useful to read some of the livelier scientific autobiographies. Good ones are Gregory Bateson's *Advice to a Young Scientist*, Freeman Dyson's *Disturbing the Universe*, Richard Feynmann's *Surely You Are Joking, Mr. Feyn-*

mann!, George Hardy's *A Mathematician's Apology*, and Jim Watson's *The Double Helix*.

A month or two after you've completed a project such as a thesis, you will probably find that it looks utterly worthless. This backlash effect is the result of being bored and burned-out on the problem, and of being able to see in retrospect that it could have been done better—which is always the case. Don't take this feeling seriously. You'll find that when you look back at it a year or two later, after it is less familiar, you'll think "Hey! That's pretty clever! Nice piece of work!"

Endnote

This document incorporates ideas, text, and comments from Phil Agre, Jonathan Amsterdam, Jeff Anton, Alan Bawden, Danny Bobrow, Kaaren Bock, Jennifer Brooks, Rod Brooks, David Chapman, Jim Davis, Bruce Donald, Ken Forbus, Eric Grimson, Ken Haase, Dan Huttenlocher, Leslie Kaelbling, Mike Lowry, Patrick Sobalvarro, Jeff Shrager, Daniel Weise, and Ramin Zabih. We'd like to thank all the people who gave us the wisdom that we pass on in this document (and which, incidentally, got us through our theses), especially our advisors.

Some of the ideas herein were lifted from "On Being a Researcher" by John Backus and "How to Get a PhD in AI," by Alan Bundy, Ben du Boulay, Jim Howe, and Gordon Plotkin.