Genifer
– an artificial general intelligence

Yan King Yin (general.intelligence@Gmail.com)

with input from:
Abram Demski
Ben Goertzel
Martin Magnusson
Russell Wallace
Pei Wang

Contents

0 Preface

1. This book is a perpetual draft. Some people get upset that it is very unpolished and contains ideas that I often retract later. I find the pros of doing this out-weight the cons: It allows us to develop our theory much more rapidly. Very often the details I glossed over are superceded by higher-level design changes and the previous details recede to much less importance. The con is that this book may not be up to scholarly standards.

2. My *personal* top priority is to obtain life extension if and when the technology is available. I guess:
   if sufficiently many people want technology X,
   and X can be achieved at a cost affordable by those people,
   then X is likely to happen.
   I estimate my natural death would be around 2048, so for me AGI must happen before 2023, to allow 25 years of AGI diffusion (just to be safe!) That means I have ∼13 years to build an AGI. I don't mean to sound selfish, this is just to let people understand my priorities and motives better.

3. The source code of Genifer is hosted on LaunchPad and this older website contains additional information not transferred to this book yet. Also feel free to contact me!

4. I'm not a "terrorist" or anti-American. I'll elaborate on this more when I have time.

— YKY

1 Introduction

I am an enthusiast, but not a crank in the sense that I have some pet theories as to the proper construction of a flying machine. I wish to avail myself of all that is already known and then, if possible, add my mite to help on the future worker who will attain final success.
— Wilbur Wright

Everything should be made as simple as possible, but no simpler.
— Albert Einstein (rephrased)

An inventor is simply a fellow who doesn't take his education too seriously.
— Charles Kettering

When a subject becomes totally obsolete we make it a required course.
— Peter Drucker

This book describes a theory of AGI (Artificial General Intelligence) [Goertzel and Pennachin, 2007] that is still being developed. I think the AGI problem can be decomposed into ∼10 computational issues and we will somehow integrate them together:

knowledge representation
fuzzy-probabilistic logic
deduction
abductive reasoning
inductive learning
pattern recognition / categorization
belief revision
memory organization
natural language
sensory processing

My approach is predominantly logic-based, but it also employs redundancy in knowledge representation, including sub-symbolic knowledge, and therefore is somewhat like neural networks. Also, it may merge with neural networks at the sensory level. This approach can be called "neo-classical AI".

1.1 Some background of AI

The word "logic" comes from *logos* which can mean "word", "thought", or "reason". The study of logic is the study of the **mechanisms of thinking**. Aristotle (ca 350BC) is often credited with the first substantial study of logic, with a focus on syllogisms. Our current system of logics was developed by people such as De Morgan (1840s), Boole (1840s), Frege (1879, *Begriffsschrift*), Russell and Whitehead (1903, *Principia Mathematica*), and others (including Liebniz (1670s, "algebra of thought") whose work remain undiscovered till the 1880s).

Logic-based AI
Thus it is not so surprising that the design of AI can be based on logic. John McCarthy (who coined the term "AI" in a 1956 Darthmouth conference) was first to propose the use of **formal logic** in AI. Herbrand (1908-1931) created a basis of provability in predicate logic. In 1965, Robinson discovered the **resolution** method for logical inference, which enabled the creation of **logic programming** and the language Prolog (Kowalski and Colmerauer, 1970s).

Connectionism
In 1943, McCulloch and Pitts formulated a formal model of neurons, leading to Rosenblatt's Perceptron (1957), and later the computational paradigm of artificial neural networks (ANNs). In the 1980s there was a resurgence of interests in ANNs and connectionism due to the invention of the backpropagation algorithm for multilayer perceptrons. At this point it was recognized that connectionism has the strengths of **robustness** and **graded response**, in contrast to logic-based AI's **brittleness** and **bivalence**.

Recent trends In the 1990s there was rapid development in **statistical learning**. It was then realized that ANN learning algorithms are a special case of statistical learning. Recent development in AI is distanced from "GOFAI" (Good Old-Fashioned AI) by their use of statistical learning, **sub-symbolic representations**, and **optimization methods** (computational intelligence). Computational intelligence includes new paradigms such as evolutionary computing (drawing inspiration from sexual reproduction) and swarm intelligence (drawing from social interactions).

1.2 Why logic?

I started designing AGI using the neural network approach for a few years, until I discovered some fundamental difficulties in the NN approach, so I switched to a more logic-based one[1].

Logic-based AI went out of vogue beginning in the 1980s because of the advent of connectionism and later statistical learning methods. As a result, many researchers nowadays are unfamiliar with even the basics of logic-based AI. In order to understand my approach it is extremely important to be familiar with **first-order logic** (FOL) and to understand the difference between first-order representations and propositional ones.

Propositional vs first-order. This is a typical propositional statement:

$$p \vee q \wedge r$$

and a typical first-order statement:

$$p(X) \vee q(X, Y) \wedge r(w(Y, Z))$$

The chief distinction is the use of *variables* in first-order logic, which greatly increases its expressiveness.

The vast majority of statistical learning literature assumes that the data is represented by points in a high-dimensional space. I call this the "spatial" approach which includes methods like nearest neighbor, support vector machines (SVMs), principal component analysis (PCA), and artificial neural networks (ANNs). *Spatial datasets are equivalent to propositional representations.* First-order logic is a symbolic or relational[2] approach which is qualitatively very different. There is strong evidence that some datasets can be easily learned by relational methods but are very difficult, if not impossible, to learn with spatial methods. ([Thornton, 1996], [Thornton, 2000].[3] provides an interesting and detailed analysis of this issue.)

To put it more bluntly, I suspect that propositional approaches are rather useless in the logic-based AI framework (but I'd be pleasantly surprised to learn otherwise).

Notice in the figure below, that all **spatial classifiers** work by "chopping" the space of data points (shown in 2D here) into various regions with the use of hyper-planes (as a line in 2D) or some curved boundaries. This is very different from how first-order logic classifies data.

[1] Though my approach is not purely logic-based.

[2] Relational representations are a subclass of first-order representations that do not allow functors and structured terms.

[3] He demonstrated this with the example of a "checkerboard" pattern of 0's and 1's that can be easily learned by a logical formula, but would be very difficult for a neural-network learner. This is actually the age-old debate between symbolic AI and connectionism, given a new twist in the context of machine learning.

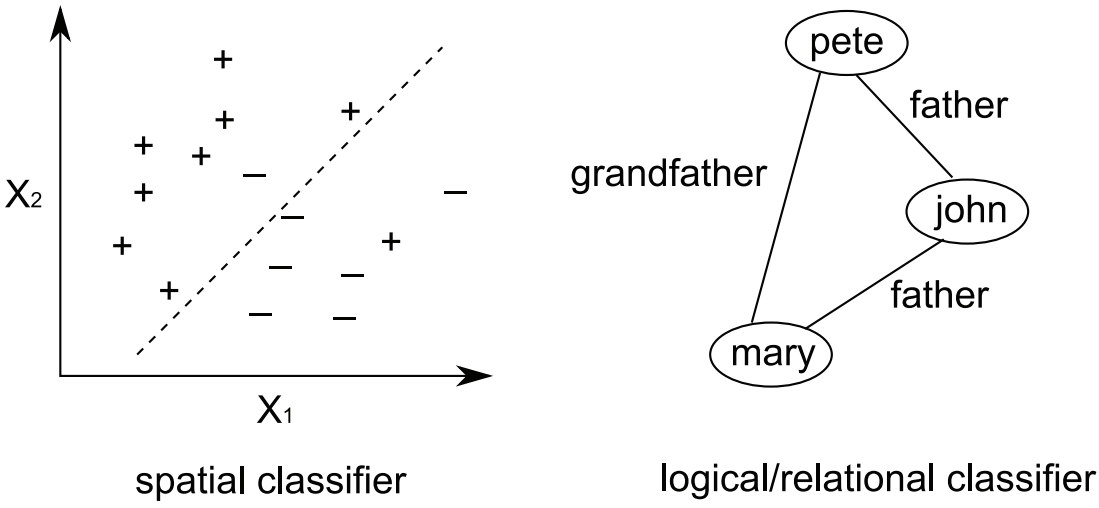


Figure 1.1: Spatial vs logical classification

To illustrate this with an example: Let's think of how a child (AGI or human) learns the concept of (blood) relatives. The child (Jane) would be given some examples of people around her and whether they are her relatives, eg:

father(jane, john), relative(john)
mother(jane, mary), relative(mary)
uncle(jane, pete), relative(pete)
neighbor(jane, joe), ¬relative(joe)
friend(jane, kate), ¬relative(kate)

and the task is to learn a general rule for relatives. The solution can be stated quite succinctly[4] in first-order logic:

relative(X, Y) ← parent(X, Y)
relative(X, Y) ← sibling(X, Y)
relative(X, Y) ← married(X, Y)
relative(X, Y) ← relative(X,Z), relative(Y,Z)

but it is very difficult to express in propositional logic unless we limit the domain of entities to a few people. Also, we can see that *spatial* statistical learning will fail to learn this rule because:

1. The dataset cannot be represented as numerical values in a vector space, or it could be done only very awkwardly.
2. Even when the dataset is cast in vector space, the learning algorithm can mostly learn to classify *existing* examples, but the *generalizations* would be wrong – this is because the formulae in first order logic can entail discrete examples that are not necessarily located in a localized region in the numerical space. Even if you carve the space into ridiculously complex regions, the next example would still be an exception because "spatial compactness" is simply absent in the underlying concept.
3. The child's world typically has very few people in it, yet she is able to learn the concept. In ANNs and statistical learning, the sample size is typically at least 100s, but logic-based learning can learn the concept with just a few examples.

Although first-order representations can be converted to propositional ones via the process of propositionalization, such algorithms cost exponential time and space. This is not difficult to see: FOL allows us to express knowledge very succinctly. There are some techniques that ameliorate the combinatorial explosion, such as partial instantiation ([Chandru and Hooker, 1999]) or sparse matrix ([Domingos, Poon, and Sumner, 2008]). But I still think it is easier and more intuitive to working on a FOL KB directly, especially for AGI.

And, despite propositionalization, the class of spatial statistical learning techniques still seem to be unsuitable for logic-based AGI because propositionalization does not cure the fundamental lack of "spatial compactness" in logical structures that I pointed out above. (After propositionalization, some fast propositional SATisfiability algorithms can be invoked, but they are still qualitatively different from the spatial learning algorithms.) As a

[4] This solution is not entirely correct, as relatedness can grow unbounded and everyone would be ultimately blood-related. Perhaps this problem can be resolved by fuzziness and other mechanisms such as non-monotonicity, but my point here is to show that the situation for propositional representation is even worse, as the problem appears insurmountable in that case.

result of this, my current strategy is to focus on algorithms specifically for FOL.

{ Update: [Gartner, 2008] describes a general method to construct kernels for (first-order and higher-order) logic formulae, based on a representation of logic by typed lambda calculus formulated by [Lloyd, 2003]. This may be a bridge between logical and spatial methods, but I have not looked into it in detail (It involves the use of a "matching kernel" that seems to have bad properties). Also, the theory of topoi and categories may offer a way to construct morphisms between models of logic theory and geometric spaces (cf Joseph Goguen's unified concept theory and theory of institutions). Also, [Muggleton, Lodhi, Amini, and Sternberg, 2005] developed a support-vector ILP method. }

{ Some new techniques have been developed to lift neural networks to first-order representations, but I have not examined them in detail (eg, [Garcez, Lamb, and Gabbay, 2009]). }

1.3 Why not neural network?

There are several reasons why I think the NN approach may be less promising:

1. First-order logic is a more powerful representation scheme than (feed-forward) neural networks (§1.2), whereas dynamic neural networks are very difficult to work with.

2. A neuron is "fixed" within a network and cannot "move around", which seems to make it difficult to perform invariant pattern recognition (eg, translational, rotational, and scale- invariance in vision). The brain has to use neurons due to biological constraints, but it seems more effective to use other pattern matching methods on von Neumann machines. (Scale invariance is particularly difficult for ANNs, see [Muresan, 2004].)

3. Neural learning is slower and require a larger amount of examples. Logic-based learning is coarse-grained and thus require fewer examples to induce the correct representation, sometimes as few as 1 example.

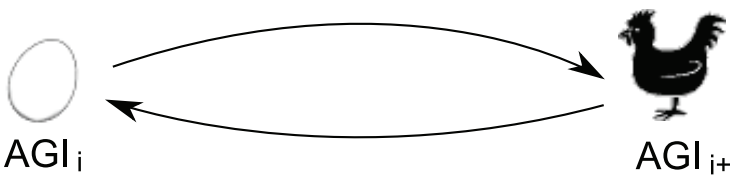
4. As Ben Goertzel pointed out some years ago, a network of redundant propositions can be reduced to a minimum number of non-redundant propositions, without loss of information; the only thing that is lost is *fault tolerance*.

However, neural networks may be used for handling low-level vision, especially at the feature-extraction level.

1.4 Recursive self-improvement

RSI refers to the ability of an AGI to reprogram itself. Some authors predict that the RSI point will trigger the Technological Singularity (eg [Kurzweil, 2005]). I think the way to reach the RSI point with the least amount of efforts would be to build an automatic program synthesis tool that accepts natural-language commands or goal specifications. From then on, we can use this tool to rewrite the tool itself and to evolve it (semi-automatically) into a full AGI system.

1.5 Chicken-and-egg problem



Anyone who has thought about AGI long enough will be aware of this problem: The idea is to use a "program evolver" that takes an input program Π_i and improves it to Π_{i+1} according to some user-specifications. We put the evolver through itself, thus building up its intelligence recursively without doing any programming (except for the initial evolver).

This idea (at least naively) does not work because the initial evolver needs to have very good background knowledge about programming or else it cannot perform its job in reasonable time. §2.5 discusses how to make it feasible.

1.6 Natural reasoning

Other names for natural reasoning are: common-sense reasoning, human-like reasoning, informal reasoning.

Natural reasoning is an extension of logical reasoning with:
1. ability to recognize natural concepts (which I posit requires fuzzy pattern recognition)
2. ability to use metaphors, similes, analogies, and similarity-based reasoning

An example of natural reasoning is:

> *Suppose I need to write a program to "break English sentences into words". I'd need to declare a function to do this. What would be the input and output of this function?*

Note that in the above reasoning, I think of the function as a "box" with something that goes in and something out.

Natural reasoning is required to turn informal, natural-language statements into formal statements. This is especially important to formal program synthesis.

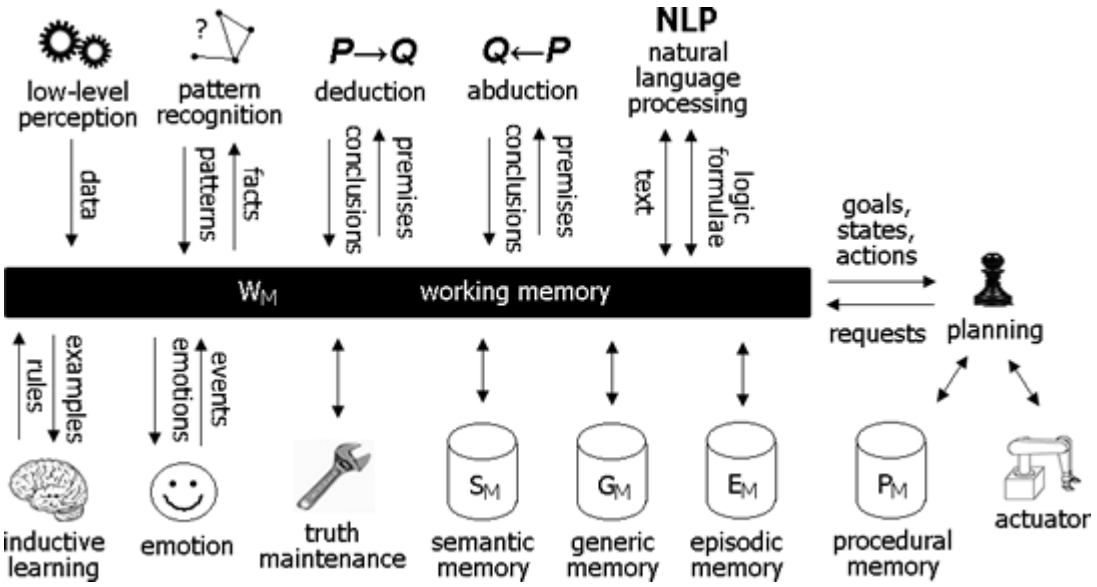
2 Architecture

We will first look at various basic architectures and then try to decide on a final design.

2.1 Logical reasoner (blackboard architecture)

This is an older design I explored around 2006, chiefly to put together several logic-based algorithms on a blackboard. The following page is a schematic diagram showing various modules of the **logical sub-system**.

My intuition is that such a logical sub-system is essential to any AGI, but it should be built on top of a more basic, *procedural* layer (cf §12.1, "Procedural subsumes Declarative").



Here is a simple top-level algorithm to process incoming sensory events:

Algorithm 1: Top-level sensory processing
Input: raw sensory input
Output: updated KB

(1) Upon the arrival of raw sensory input, perform **pattern recognition** (§6) which is the same as forward-chaining. (NL input can bypass this step.) The result will be a set of *new facts*.
(2) Perform **consistency check** (§7.3) on the new facts against the KB
(3) If a new fact is inconsistent with the current KB, invoke **conflict resolution** (§7.4)
(4) Perform goal-less **forward chaining** (§5.3) on the new facts, and put the results in Working Memory. Thus WM will be aware of the new facts' immediate consequences.
(5) Perform **abduction** (§5.4) on the new facts, so WM will make appropriate assumptions to account for them.
(6) Invoke the **inductive learner** (§9.2), ie, try to compress the new facts + KB.

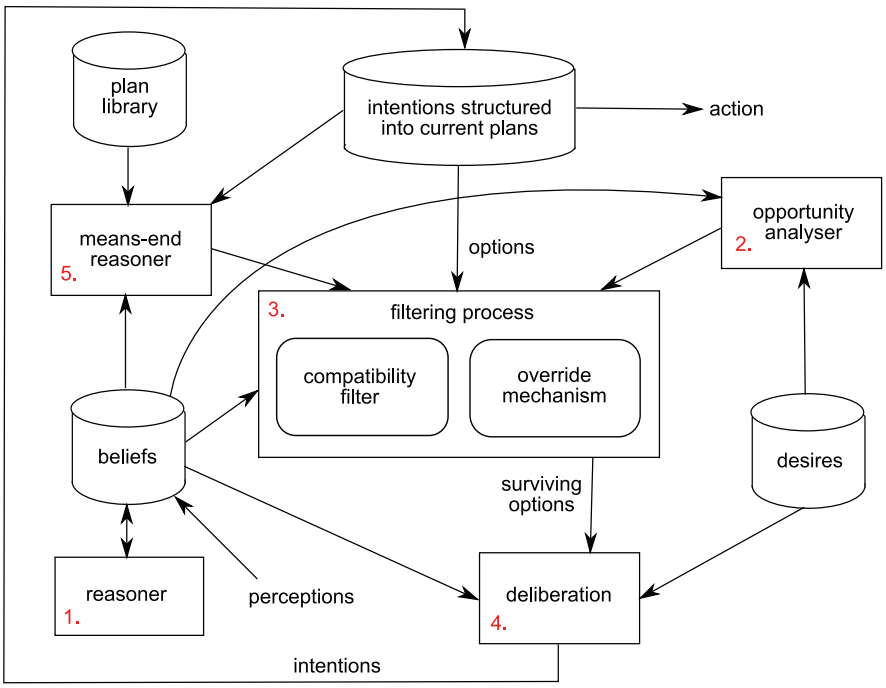


Figure 2.1: schematic diagram of the BDI architecture

The abduction algorithm is almost identical to the consistency-check algorithm, so they may be merged. The abduction algorithm is also very similar to the induction algorithm (cf §5.4), so all 3 may be merged.

2.2 BDI architecture

BDI (**belief-desire-intention**) is one of the most successful agent architectures ever proposed. It was first investigated by Michael Bratman as a theory of human practical reasoning ([Bratman, 1987]), and later formulated as an agent architecture ([Bratman, Israel, and Pollack, 1988]). The main concern of BDI is how to plan in the face of a *dynamic environment* and with *bounded resources*.

Key to BDI is the idea of **commitment**. Once an agent commits to a plan, the plan will constrain means-end reasoning, thus making planning processes more efficient.

B: Beliefs are the contents of the KB.
D: Desires are unconstrained; they need not be achievable or even consistent (eg, one can desire smoking and good health at the same time).
I: Intentions are goals that the agent commits to; they are believed to be achievable and must be consistent.

Figure 2.1 is a schematic diagram of BDI. The key processing steps are:

1. This is the **Knowledge / Declarative Sub-system**, which we will consider at length in the rest of the book.
2. Changes in the environment cause changes in beliefs, which in turn reveal new possibilities to satisfy desires. Thus the **Opportunity Analyzer** tries to propose new **options**.
3. Once the options are produced (either by the Opportunity Analyzer or the Means-end Reasoner), they are subject to **filtering**. The **Compatibility Filter** checks if options are compatible with existing plans.
4. Surviving options are weighted against each other by the **Deliberation Process**, which produces intentions to be incorporated as plans.
5. The **Means-end Reasoner** searches for the next available options, taking into consideration the current plans.

2.3 Evolutionary architecture

{ TO-DO: Hayek is not the correct designation for this architecture }

Hayek is an artificial economy proposed by Eric Baum ([Baum, 2004]) as an AGI architecture. A slight variant of Hayek can be described as follows:

1. The AGI is composed of a large number of small programs.
2. **Cooperativity**. The programs may call each other.
3. **Persistent memories**. Individual programs can remember things across time slices.
4. A **meta-controller** runs these programs according to some schedule, allotting each program a time slice.
5. **Credit assignment**: If the program answers a question correctly or performs a good action (judged by some external critic), the meta-controller will credit the programs that have contributed to the result.
6. Human programmers may contribute to this pool of programs.

A high-level programming language (such as Lisp) seems to be more suitable for this purpose.

Note that even very complex algorithms can be implemented in this architecture. For example, a best-first search algorithm can remember its search state in an external memory store. Then it just waits for its time-slice to resume searching. Thus human programmers can seed the artificial economy with highly competent programs.

2.4 Decision-theoretic architecture

This is mathematically more elegant than the BDI architecture.

Some important elements that should be present in the architecture:

1. **percepts** are raw (unprocessed) sensory events
2. **beliefs** are the contents of the KB
3. **states** — all possible *perceived* states of the environment, including the current state. States are special terms in the logic.
4. **actions**
5. **utility function** — a function $U : \{state\} \to \mathbb{R}$. Utilities may be defined implicitly, though.

This architecture may depend on a **logical sub-system** that:

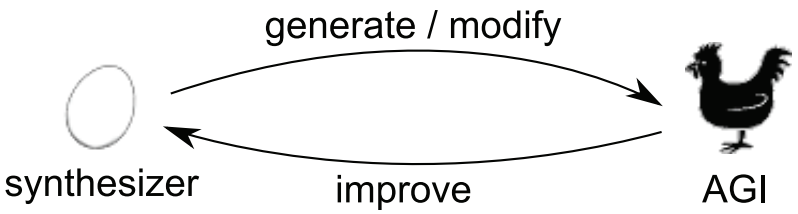
1. recognizes environmental states (eg *"the apple is on the table"*) from raw percepts (eg pixel values from a camera)
2. recognizes possible actions (eg *"I can put the apple on the plate"*)

The goal is to maximize expected utility. Planning becomes a discrete optimization problem in this setting.

Reinforcement learning can be naturally included in this architecture.

2.5 Self-programming architecture

Consider this variant of the chicken-and-egg problem (§1.5), where we distinguish between the **Synthesizer** and the AGI:



Currently we know the following program-synthesis paradigms:

1. human programming
2. natural reasoning (NR), ie common-sense / human-like reasoning
3. automated theorem proving (ATP)
4. stochastic local search (SLS), including evolutionary algorithms (EA)
5. reinforcement learning (RL)

We will consider each paradigm in turn.

A key question is how an initially-dumb AGI can *improve* the performance of the Synthesizer, even slightly.

1. Human programming (ie, brute-force software engineering without RSI)

The problem with this approach is that it does not exploit the advantages of machine program synthesis, and thus is likely to be sub-optimal.

2. Natural reasoning (cf §1.6)

(a) The problem is that NR does not yet exist, but it is a *sine-qua-non desideratum* of AGI.
(b) So the question is how to enable an initially weak form of NR to synthesize programs. One possibility is to use **creativity**: the AGI will generate programs by partly reasoning and partly random acting.
(c) An agent with NR is likely to have RL as well.
(d) An NR agent can invoke ATP.
(e) An NR agent can self-program (without invoking ATP), simply by **deductive planning** (§12.6).

3. ATP

(a) It requires *formal* specifications of programs (or sub-routines), which are often very hard to write for humans.
(b) Search in proof space can be very slow.
(c) There already exists ATP-based program synthesis software, which can be used externally.

4. Stochastic local search

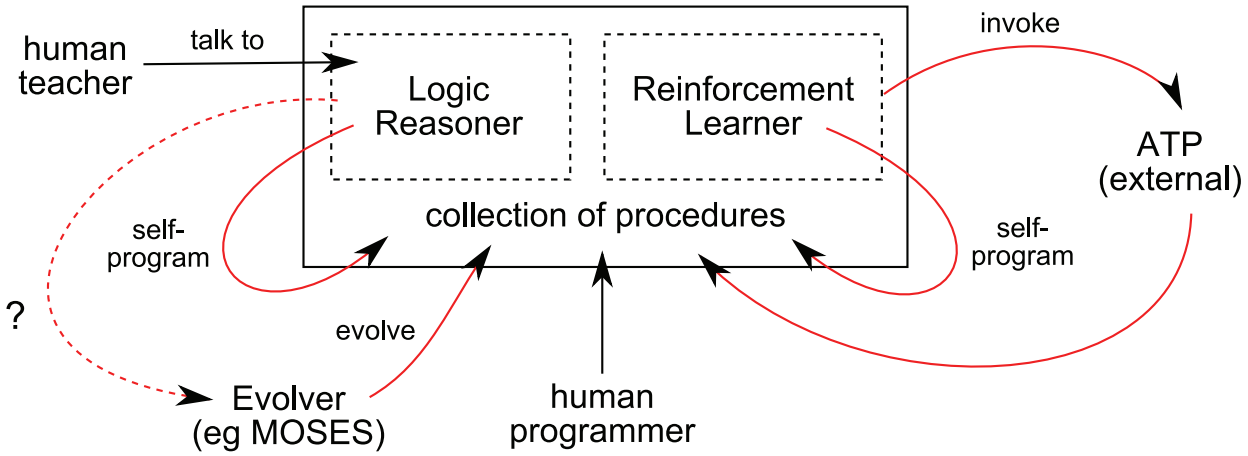
(a) Is based on generate-and-test.
(b) "Generate" is random and the program space is typically huge.
(c) "Test" is often slow because it requires the entire AGI to answer a large number of benchmark queries. A possible remedy is to specify input-out benchmarks for AGI *components*, but then the burden is on humans to design the modular architecture.
(d) A large population of relatively low-score programs has to be maintained, in order to allow sufficient time for cooperativity to evolve. For this reason I suspect that the evolutionary approach is rather slow.
(e) 2 further problems are: How can an initially-dumb AGI improve the stochastic searcher?[1] And how can this transfer of rationality from the AGI to the Synthesizer be coded once and for all?

5. Reinforcement learning

(a) The goal of RL is to learn policies of how to act in certain environmental states. It can synthesize programs if the environment is that of programming.
(b) RL can be augmented with logical reasoning.
(c) RL can implement the idea of expected utility maximization.

Combining all methods

Apparently, we can combine all the above methodologies without conflict:



The question is to choose the *most effective* method of AGI self-programming (red lines). My intuition is that the evolutionary method is much less efficient than the other 3 (it is easier to explain to children to do something than to train animals by reinforcement, which is in turn easier than breeding animals for the innate ability to do that thing), provided that some rudimentary ability of reasoning is present.

Maybe the most cost-effective method (from the human labor perspective) is to combine RL with logical

[1] In MOSES (cite) an attempt is made at "representation building".

reasoning and deductive planning — for instance, to allow RL to invoke logic or vice versa.

3 Knowledge representation

3.1 Introduction

What would be a good knowledge representation scheme for AGI? We need to understand that there is no single right answer to this question. An AGI uses a KR structure to *represent* the external world, and this KR structure is built with limited computational resources. As such, it must be an approximation of the world. This means we have much freedom in the choice of KR.

My choice is based on classical logic, especially first-order logic (FOL) because of its well-established status in AI research. Also, FOL is easy for myself and others to understand.

A common misconception is: "How can complex ideas such as 'John loves Mary' be reduced to logic formulae like *loves(john,mary)*?" One school of thought (see eg [Johnson-Laird, 1983]) posits that human reasoning is based on "mental models", but it is unclear how exactly they can be constructed. My view of logic-based AI is that of using logic (or "relational formalisms") as a *computational structure* for *constructing* mental models. It does not mean that logical formulae in an AGI correspond to "Truths" in the real world.

TO-DO: Typed or untyped logic?

3.2 Reification

TO-DO: explain what is reification, how it is represented.

3.3 Composition functor

3.4 Rus logical form

3.5 Representing time

As Einstein would have said, the representation scheme for space and time should be fundamentally the same. As I have developed a vision theory (§15), I think temporal representations can follow a similar scheme. OpenCog (http://www.opencog.org) is an AGI project more focused on embodiment, so we can also share their KR scheme.

3.6 Assumptions and counterfactuals

How to make assumptions during inference? "Assuming mom is at home, I call her phone number".

Example of a counterfactual conditional: "If Oswald did not kill kennedy, someone else would have".

3.7 Contexts

An excellent survey of contexts in logic-based AI is [Akman, Surav, and Giunchiglia, 1996]. The book [Sowa, 2000], Chapter 5, is also excellent and contains additional insights about contexts.

4 Logic

An approximate answer to the right question is better than a precise answer to the wrong question. — John Tukey (rephrased)

Abstract. $\mathcal{P}(\mathcal{Z})\mathcal{C}$-logic is a hybrid logic combining binary ($\mathcal{B}$), probabilistic ($\mathcal{P}$), fuzzy ($\mathcal{Z}$), and confidence ($\mathcal{C}$) reasoning [1]. It is specially designed for AGI (Artificial General Intelligence) [Goertzel and Pennachin, 2007] and common-sense reasoning. Emphasis is put on simplicity and efficiency of inference, which is critical to the development of complex algorithms upon this logic. Unlike traditional fuzzy or probabilistic logics, $\mathcal{P}(\mathcal{Z})$-logic is algebraic and truth-functional (thus abolishing the implication operator), which makes inference very efficient (analogous to Prolog's SLD resolution); this is achieved at the cost of approximating probabilistic inference.

* A note on the examples used in this book: some of them are politically incorrect or somewhat embarrassing. I prefer examples that are simple, realistic, and relevant to human emotions because they help us think more clearly (cf the Wason selection task). Usually I just choose the most obvious examples that come to mind.

Reasoning under uncertainty is a vast and nightmarishly complex topic in AI. Simon Parsons's book [Parsons, 2001] contains a very good survey of uncertain reasoning, but even that is not exhaustive. We may look at the following taxonomy of "ignorance" proposed by [Bosc and Prade, 1997]:

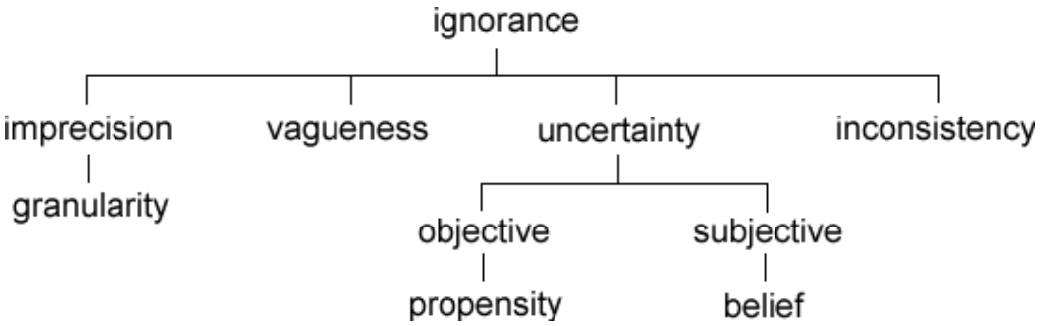


Figure 4.1: taxonomy of ignorance

At first blush, classical logic appears to be sufficient for AGI. Some AGI designers prefer to use crisp logic as the base and plan to build $\mathcal{P}$ and $\mathcal{Z}$ upon crisp logic. But this may be a sign of not facing problems and not having sufficient understanding of fuzzy-probabilistic logics. To represent $\mathcal{P}$ and $\mathcal{Z}$ logic *in* crisp logic is like writing an entire $\mathcal{P}$-$\mathcal{Z}$ inference engine in Prolog, complicated by the fact that the crisp logic would be somewhat different from Prolog and so may be even harder to program in. Also, the "truths" known by the AGI will then be different from the truths as represented in the crisp logic — the former would be "floating" above the latter; This creates unnecessary indirectness. I will give other reasons in §4.3.2 and 4.4.1.

Why not include other uncertainty measures besides $\mathcal{P}$ and $\mathcal{Z}$?

There are other theories of uncertainty, such as possibility, belief functions, and rough sets. The reason I chose $\mathcal{P}$ and $\mathcal{Z}$ is because they are simple and best understood. There has been attempts to create systems where the user can create a flexible number of uncertainty measures, but one problem of such systems has been pointed out in [Parsons, 2001]: If we have 3 uncertainty measures, say "cloud", "mist", and "fog", there would be a need to provide mixed inference rules for "cloud-mist", "cloud-fog" etc, a total of n^2 possibilities. I have worked out the combination of $\mathcal{B}$, $\mathcal{P}$ and $\mathcal{Z}$ and found that it involved considerable efforts.

4.1 Shortcomings of the current logic

1. **Binary vs $\mathcal{P}/\mathcal{Z}$.** It may be beneficial to have binary logic alongside $\mathcal{P}/\mathcal{Z}$ logic. $\mathcal{P}/\mathcal{Z}$ reasoning is suitable for common-sense concepts, whereas binary reasoning is good for *programmatic* [2] or computational reasons. However, one unsolved problem is that many common-sense concepts appear to be binary but are fuzzy upon close inspection (eg male/female, dead/alive). The question is how to let binary and fuzzy concepts coexist in the same logic.

2. **First-order vs Higher-order logic.** FOL is insufficient for AGI, see §4.2.1.

[1] I use $\mathcal{Z}$ instead of $\mathcal{F}$ to denote fuzziness partly in recognition of Lotfi Zadeh's contributions, and also because F and f are often used for the CDF and PDF in probability theory, which may cause confusion when probabilities and fuzziness are used together.

[2] Ie, binary logic makes programming easier

4.2 $\mathcal{B}$: binary logic

My intuition is that common-sense reasoning involves mainly $\mathcal{B}$, followed by $\mathcal{Z}$, and $\mathcal{P}$ is relatively rare.

4.2.1 First-order logic sufficient?

Let us admit outright that FOL as a KR scheme is insufficient for common-sense reasoning and thus AGI. But I also have the impression that second-order logic would be sufficient for most common-sense reasoning purposes.

With some tricks (such as reification, §3.2), FOL may be able to deal with second-order reasoning.

Another trick is that the FOL reasoner can "escape" to a meta-reasoner when higher-order constructs are needed. This will be taken up on the chapter on meta-reasoning, §8.

It seems much easier to deal with higher-order issues using these tricks than to use HOL as the base logic.

4.3 $\mathcal{Z}$: fuzziness

4.3.1 Vague phenomena

There seems to be 3 types of concepts (= predicates).

The first type is **discrete** in nature, for example[3]:

sex of a person $\in$ { male, female, hermaphrodite, trans-sexual }
marital status $\in$ { single, engaged, married, divorced }

The second type represents **numerical** measures, for example:

height weight age

The third type is not strictly numerical, but is often associated with "**gradedness**":

beautiful	ugly
intelligent	dumb
simple	complex
interesting	boring
good	bad
easy	difficult
friendly	hostile
clear	unclear

There is no doubt that these predicates can be modified with degrees like "very" or "slightly". But we do not know of any exact measures of their degrees. For example, IQ has been proposed as an inexact measure of intelligence. Most people would agree that there is some general consensus as to what is intelligent. Such concepts exhibit *vague phenomena*, which are characterized by:

1. borderline cases
2. lack of sharp boundaries
3. susceptible to sorites paradox[4]
4. open-texture (ie, if x is a borderline case, one can assert either $Q(x)$ or $\neg Q(x)$ in different contexts)

In philosophical logic, there are a number of theories of vagueness (see eg, [Graff and Williamson, 2002], [Keefe, 2000], [Shapiro, 2006]). $\mathcal{Z}$ logic is a kind of Degree Theory ([Edgington, 1992], [Sainsbury, 1986]), that uses numerical truth values to represent vagueness. There are other non-numerical theories such as Epistemicism ([Campbell, 1974], [Williamson, 1994]), and Supervaluation ([Fine, 1975], [Keefe, 2000]).

4.3.2 Why vagueness is needed for AGI

One of the critical capabilities of AGI is (self-)programming. Also, it is imperative to be able to instruct an AGI to write programs according to *natural language* specifications and with robust common-sense background

[3] I will later argue that even these concepts are not completely binary.
[4] A solution to the sorites paradox is given in [Bergmann, 2008] p268-282, using fuzzy logic and the notion of *decaying validity*.

knowledge. common-sense reasoning and natural language understanding depend on the use of vague concepts.[5] Look at any (technical or non-technical) natural-language text, and one finds that (explicit or implicit) vague concepts are ubiquitous. It seems to be occur even more frequently than the (explicit or implicit) use of probabilities.

Ubiquity of vagueness in common-sense reasoning. Some examples are:
- Time: *Mother died a few days ago*
- Space: *One bird flew over the cuckoo's nest* (lacking exact boundaries)
- Physics of liquids: *A liquid in bulk and at rest has a horizontal surface*

(Is soup a liquid? "In bulk" is a fuzzy concept. Is the sea at rest? Surface tension can cause surface to curve)
- Physics of solids: *A solid object cannot go from inside to outside a closed box*

("Solid object" is a fuzzy concept (eg a block of ice, a cat, a bomb). A card box may have small holes. The lid may be slightly ajar.)

4.3.3 Why numerical vagueness?

The controversy is whether vagueness should be managed **quantitatively** (such as $\mathcal{Z}$) or **qualitatively**.

One objection is that fuzzy logic can sometimes lead to unsound conclusions. This problem is discussed in §5.2.1 on nonmonotonic reasoning.

Having numerical vagueness allows us to:
1. Represent quantitative rules. For example:
   smart ← eloquent (the more articulate the smarter)
   ← humorous (the more humorous the smarter)
   ← insightful (the more insightful the smarter)
   ← creative (the more creative the smarter)
   ← etc...
2. Add up graded factors. For example:
   "John is eloquent, humorous, insightful and creative" → 0.9 smart
   "John is humorous and nothing else" → 0.6 smart
3. Accrue contributing factors of a concept over a long period of time:
   "From my long experience with John, he is 0.9 smart".

On the other hand, if we do not use numerical vagueness, we face these problems:
1. Failure to recognize "partial" concepts. For example:
   "John is 0.7 smart", or
   "tomato is 0.7 a fruit"
2. Conversely, failure to ignore "very weak" partial concepts.
3. Each statement must be attached with many qualifications, and they keep accumulating. Each rule would have to recognize a large "exception set".

The solution I adopt is to use both qualitative and quantitative information, by representing facts *redundantly* (§5.2.1). For example:
   "John is smart", $z = 0.7$ (quantitative)
   "But he is only penny wise" (qualitative)

Lastly, what if we don't need the precision of numerical vagueness in some situations? For example:
   "Is John really that smart?" "Not quite." (we don't know the exact degree of smartness)
or
   "John is slightly smarter than Peter." (but we don't know the exact difference)
In my theory, these cases can be handled by combining $\mathcal{P}$ and $\mathcal{Z}$.

4.3.4 Semantics of Z

There is some confusion about the interpretation of fuzziness, which I will try to clarify here. I am influenced by Pei Wang's ideas [Wang, 2006].

[5] Vague concepts are not required for formal reasoning tasks such as formal mathematics and programming according to formal specifications.

$\mathcal{Z}$ is a measure of *degree* or *gradedness*. Standard fuzzy theory employs the "membership function" to represent *the degree to which an element belongs to a class*, but there is no consensus as to how the membership functions are defined. Let's think of "the degree to which a person is smart", is it really arbitrary?

While there are no exact procedures to measure vague concepts (eg smartness), it is mandatory that a common-sense machine should possess some ways of assessing them, just like the human brain does. In my AGI this is achieved by having a comprehensive knowledgebase of rules (acquired via machine learning) that compute numerical degrees of concepts. Such rules are mini-algorithms (or "**micro-theories**") that are **emergent properties** of intelligent learning systems. They establish a *distributed and approximate consensus* of how to measure vague concepts.

From another perspective: PCA (principal component analysis) can calculate the component of a dataset that represents its greatest variance. For example, PCA may identify the male-female axis of a set of movie-goers, or the liberal-conservative axis of a group of politicians. When we map this axis to [0,1], we can get a fuzzy value of the concept male/female or liberal/conservative. This may answer the question "where do fuzzy numbers come from?" [6]

If we regard $\mathcal{Z}$ as a measure of degrees, $\mathcal{Z}$ theory is mathematically as rigorous as probability theory. $\mathcal{Z}$ is a measure of degrees just as the height H is a measure of how tall an object is. Also, the $\mathcal{Z}$-value can be inexact just as H can be inexact, but this inexactitude is modeled separately by *distributing probabilities over* $\mathcal{Z}$ (§4.6).

$\mathcal{Z}$ is not an approximation of probability; §4.3.5 gives a probabilistic interpretation of vagueness, but in practice we can treat $\mathcal{P}$ and $\mathcal{Z}$ as **orthogonal** to each other. Possibility theory defines fuzziness as a weaker form of probability (as non-additive probability), but this is not the approach of $\mathcal{Z}$ logic.

Finally I want to dispel the myth that probability is mathematically more rigorously defined than fuzziness:

probability is defined by Kolmogorov's axioms	fuzziness can be defined by similar axioms (§4.3.9)
probability follows a rigorous calculus	fuzziness follows a rigorous calculus
probability is a subjective measure that exists only in the mind	fuzziness is a subjective measure
probability can be exactly calculated for some cases, eg: dice or coin	fuzziness can be exactly calculated for some cases, eg: age, temperature
some probabilities in our mind have obscure origin, eg: predicting the outcome of an election	some fuzzy values in our mind have obscure origin, eg: which leader is more charismatic

Quantum mechanics or Heisenberg uncertainty does not prove that probabilities exist in the physical world. On the other hand, studies in chaos and complex systems reveal that macroscopic descriptions are emergent and distinct from microscopic descriptions even though the former can be reduced to the latter. Thus the recognition of fuzzy macroscopic patterns is every bit as real as the subjective use of probabilities to describe physical systems. Vague concepts are also useful in thinking about pure mathematics – for example, recognition of fuzzy similarities may assist mathematical reasoning.

4.3.5 Probabilistic interpretation of vagueness?

One way to interpret vagueness as probability is to interpret

$$Q(x);\ z = z_0$$

as

"Among all possible contexts, $Q(x)$ is true with probability z_0".

For example, if $smart(john);\ z = 0.8$ then John is smart in 80% of circumstances.

More examples:

z = 0.8	**Interpretation**
John is very smart	John is smart in 80% of circumstances
Peter is very fat	Peter is fatter in 80% of comparisons (with other folks)
Jane is very pretty	Jane is judged pretty by 80% of beholders

[6] Abram Demski suggested this to me in discussion.

There seems to be two problems with this interpretation. The first concerns the min-max calculus (§4.3.8) – if $\mathcal{Z}$ is really probability, why does it not obey the probabilistic sum-product calculus?

The second problem is very subtle, and concerns the nature of matters of degree. Consider these 2 statements:

A: "Mary is 0.8 probably ugly" ($p = 0.8$; thus a 0.2 chance of NOT ugly)

B: "Jane is 0.8 ugly" ($z = 0.8$)

Suppose John *must* find a pretty girl. If John has seen Jane and judged her to be 0.8 ugly, then there seems to be no uncertainty about it in this context (John being the judge and Jane having her present looks, etc). So if John really must find a 0.9 pretty girl then he should prefer Mary (whom he hasn't met and has $p = 0.2$ chance of being pretty).

In §4.6.1 we will consider probability distributions over fuzziness, where this problem reveals a subtle difference in the shapes of probability distributions.

4.3.6 Reference classes

The measure of a $\mathcal{Z}$ value is dependent upon its *reference class*. For example, if we want to say how "young" a person is, the reference class may be "all people" or "all tenured professors". The measure of "youngness" thus varies depending on the reference class.

Once the reference class is fixed, it seems that $\mathcal{Z}$ is *not* context-dependent. For example, we can say "John is a young man who owns an old dog". The dog is described as "old" using its own reference class (dogs), not John's reference class (humans).

4.3.7 Numerical scale of Z

Many "natural" quantities occur in the range $[0, \infty)$. For example, the height, weight, age, or ugliness of a person can theoretically range from 0 to ∞. It is unnatural to set artificial upper limits to these measures. However, it may be possible to extend these concepts to the range $(-\infty, \infty)$. For example, the age of AGI may be ~ -10 because it is not yet born. I reserve this possibility but will use $[0, \infty)$ for now.

$\mathcal{Z}$ is defined in $[0, 1]$. So we need **membership functions** to map $[0, \infty)$ to $[0, 1]$. I choose a sigmoid function with parameter ξ because it has some nice properties. We need two orientations because some concepts get more and more positive as $x \to \infty$, while others the opposite.

$$Z_1(x) = e^{-\ln 2 \cdot (x/\xi)^2} \quad \text{and} \quad Z_2(x) = 1 - e^{-\ln 2 \cdot (x/\xi)^2} \tag{4.1}$$

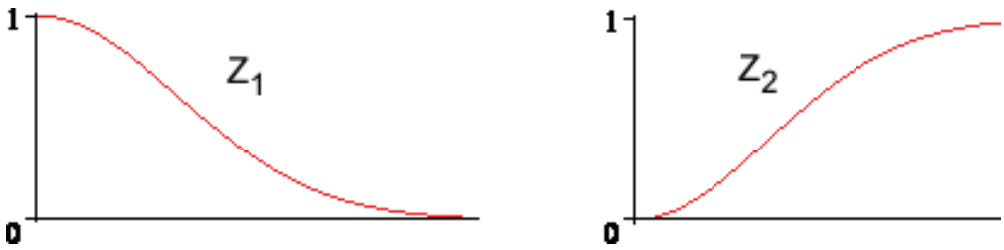


Figure 4.2: membership functions

Negation is defined as $1 - z$. As a consequence of this, for any concept, $z = 0$ means the *antithesis* of that concept. For example $Z(young) = 0$ would mean old. Another consequence is that the point at $z = 0.5$ is the *point of neutrality*, which is where a concept is neither true nor false.

Some concepts (such as "chair", "absurd") do not have natural opposites. For these concepts, $z = 0$ means the *complete absence* of the qualities in question.

Negation can cause some confusion because "not young" can mean either "middle age" or "old". The only treatment of fuzzy negation that is entirely consistent with our common-sense is to distribute probabilities over fuzziness, which will be developed after §4.6.1.

The interpretation of the parameter ξ in eqn (4.1) is that it marks the *point of neutrality* on the x-axis, for

which z=0.5. This is illustrated as follows: we map the human age x to the $\mathcal{Z}$-concept of "young", where I (subjectively) define "40 years old" as the neutral point of "young":

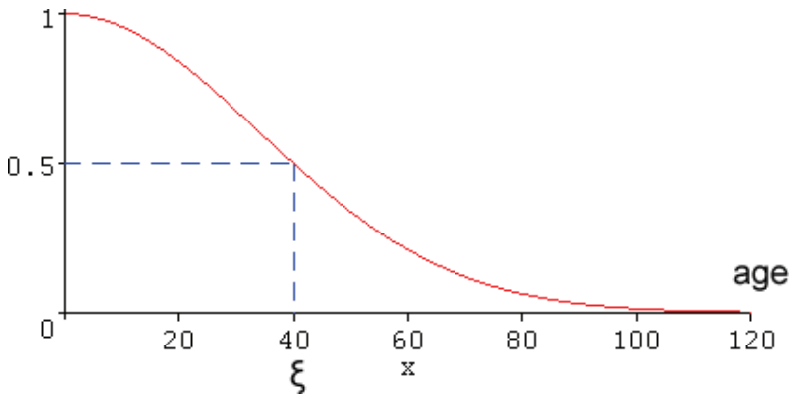


Figure 4.3: neutral point

This means, after 40, one gets more and more "not young" according to this definition.

Thus the numerical scale of $\mathcal{Z}$ is:

Table 4.1:

z	interpretation
1.0	definitely or extremely
0.9	very
0.7-0.8	moderately
0.6	slightly
0.5	neutral
0.4	slightly not
0.3-0.2	moderately not
0.1	very not
0.0	definitely not

Note: A question has been raised whether we can define opposite concepts with 2 predicates, Q^+ and Q^-, with z^+ and z^- joining at 0 in the middle. This is not entirely satisfactory because the negation of Q^+ would not *cover* Q^-, nor vice versa. Therefore, the choice of $z = 0.5$ as neutrality point is quite essential.

4.3.8 Why Z obeys min-max calculus

Probabilities obey sum-product calculus; $\mathcal{Z}$ obeys **min-max calculus** [Zadeh, 1965]. My justification for min-max is as follows:

As an example, consider the statement:

S1: *"John have had sex with 1000 women"*

but it turns out that all those women had only had cybersex with him. Most people would agree that cybersex is not quite the same as real sex (some may say it's a borderline case ($z = 0.5$)). Suppose we subjectively think that cybersex is 0.7 real sex (as a measure of degree), then what would be the "degree of truthfulness" of the statement S1 (assuming that we accept the fact that he had cybersex with 1000 women)?

If we use the sum-product calculus (as with probabilities), the answer would be 0.7^{1000} which is almost zero.

Whereas if we use the min-max calculus, the answer would be $min\{0.7, 0.7, ...\} = 0.7$. So, did John have sex with 1000 women?

answer A: "Of course not."

answer B: "Well... sort of."

My view is that the conjunction of 1000 vague events should have the same vagueness as the individual events. You may try this with other examples of graded events.

4.3.9 Axiomatic description

In summary, $\mathcal{Z}$ is the relaxation of the classical-logic view that a statement is either true or false; true, false is relaxed to [0,1].

Here is a set of axioms that describes $\mathcal{Z}$:

Z1. $z \in [0, 1]$
Z2. z varies continuously within $(0, 1)$
Z3. $z = 0.5$ is the point of neutrality
Z4. $\mathcal{Z}$ obeys min-max calculus when applied by logic conjunction and disjunction

The meaning of Z2 is yet to be clarified. For now I'd just state it informally. Also, it would be nice to formulate $\mathcal{Z}$ calculus in a way similar to Cox's postulates for probabilities, but that is not my current priority.

4.3.10 A fuzzy paradox

A common problem in fuzzy logic is concerning the truth value of statements such as "Q and not Q". It can be resolved using our understanding of $\mathcal{Z}$ negation:

Suppose $Z(tall(john)) = 0.6$ (which means that John is slightly tall)
then

$tall(john) \wedge \neg tall(john) = 0.4$ (which means this is slightly false)
$tall(john) \vee \neg tall(john) = 0.6$ (which means this is slightly true)

On the other hand, if $Z(tall(john)) = 0.4$ (which means that John is slightly short)
then

$tall(john) \wedge \neg tall(john) = 0.4$ (which means this is slightly false)
$tall(john) \vee \neg tall(john) = 0.6$ (which means this is slightly true)

All these are reasonable conclusions.

4.3.11 "Soft" min-max and concept learning

{ TO-DO: I have some doubts about this section; the argument is a bit unclear. Maybe soft min-max are unnecessary afterall... }

As an example, these are 2 exemplars of "chair" that most people consider to be typical:

Usually, the old-fashioned chair is 4-legged and is made of wood; and the office swivel chair can rotate and has wheels. These are the **features** of the exemplars stored in memory. It would be atypical for a wooden chair to have wheels and have a seat that can rotate above the 4 legs. So, even though both chairs are very typical chairs, their features cannot be exchanged freely while maintaining the same level of typicality.

features	**degree**	
wooden $\wedge$ 4-legged	1.0	typical old-fashioned chair
rotating $\wedge$ has wheels	1.0	typical office chair
has wheels $\wedge$ 4-legged	0.9	atypical chair

It seems that crisp min and max cannot represent this (but I may be mistaken about this point). Anyway, I created a "soft" version of min-max (the idea is to use z_1, z_2 as their own weights in a weighted average):

soft min (= AND)	soft max (= OR)
$z_1 \widetilde{\wedge} z_2 = \frac{z_1(1-z_1)+z_2(1-z_2)}{1-z_1+1-z_2}$	$z_1 \widetilde{\vee} z_2 = \frac{z_1 z_1 + z_2 z_2}{z_1+z_2}$

(4.2)

It can be verified that soft- min and max satisfy the boundary conditions of classical logic, provided that we make $0/0 = 1$ in the min case.

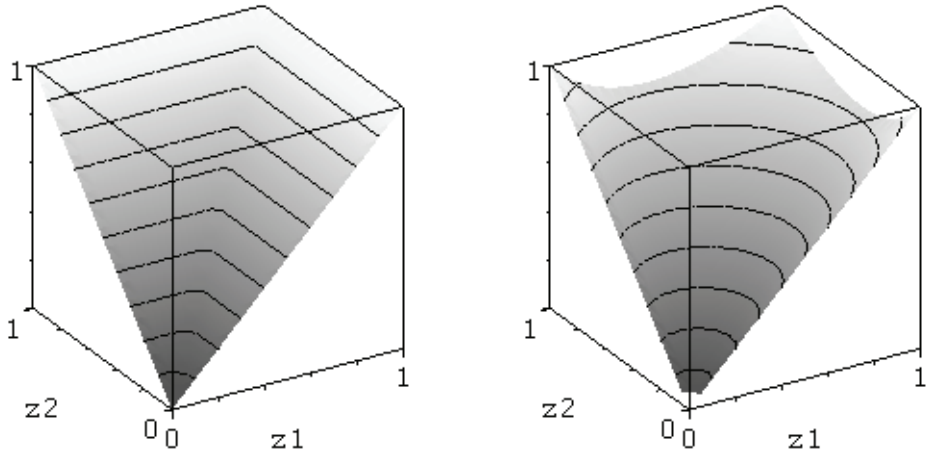


Figure 4.4: comparison of max and soft max

4.3.12 $\mathcal{Z}$ modifiers

To make $\mathcal{Z}$ logic more versatile, we need to augment it with **hedges** which correspond to natural language words like:

extremely very moderately slightly

so we can express things like:

lukewarm $\leftarrow$ moderately(warm)

obese $\leftarrow$ very(fat)

In general, we can define a $\mathcal{Z}$-modifier as a function $\Gamma : [0,1] \rightarrow [0,1]$,

$$z_0 := \Gamma(z_1) \tag{4.3}$$

We further restrict the class of Γ to make the system simpler. I suggest to use Gaussian functions with the mean z^* as a parameter, and the variance would be fixed to a certain constant. So

$$z_0 := \Gamma(z_1; z^*) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z_1-z^*)^2/2\sigma^2} \tag{4.4}$$

For example, the Γ's for "slightly" and "very" can be:

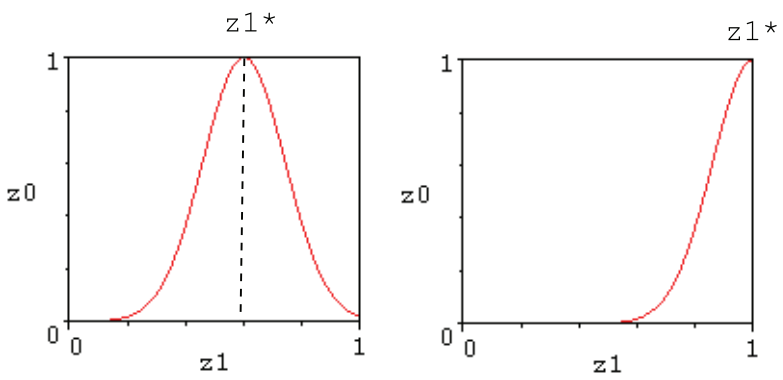


Figure 4.5: Fuzzy modifiers with $z^* = 0.6, 1.0$

We can have better control over the shapes of Γ by using other functions and having more parameters, but I suspect that such sophistication is not needed for common-sense reasoning.

For example, we can define "lukewarm" as "warm" with $z \in [0.6, 0.8]$, or:

lukewarm $\leftarrow \Gamma_{0.6}(\text{warm}) \wedge \Gamma_{0.8}(\text{warm})$

using 2 $Gamma$'s with fixed variances. The result is the blue curve on the left. We get the interval [0.6,0.8] by taking > 0.5 as true, and thus "lukewarm" would be a binary predicate.

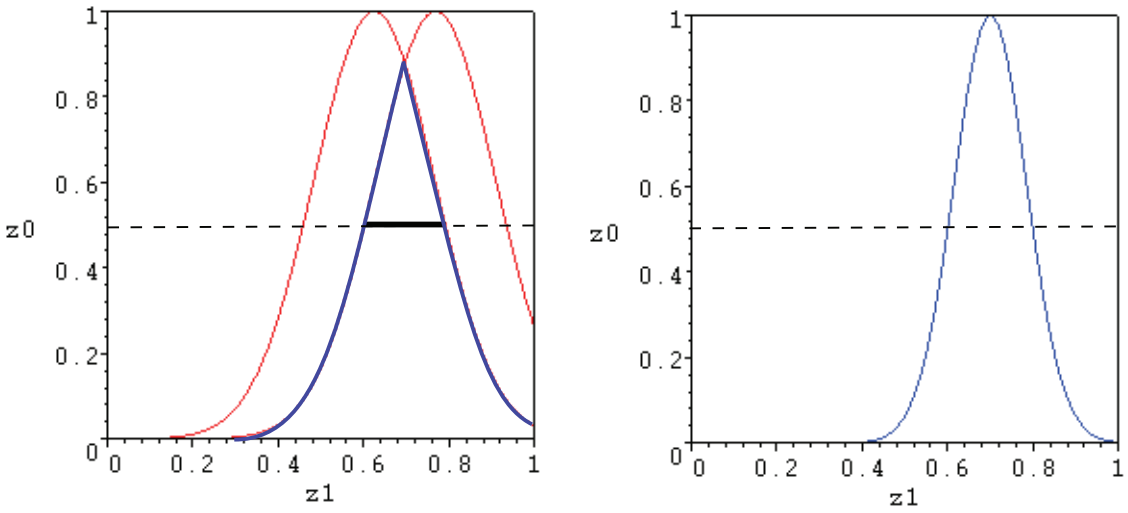


Figure 4.6: Two representations of "lukewarm"

On the other hand, if we use a tailored Γ to represent "lukewarm" on the right, it would be a $\mathcal{Z}$-predicate with continuous values like "slightly lukewarm" and "very lukewarm". This seems to be unnecessarily sophisticated.

{ TO-DO: One problem with this method is that the predicate "lukewarm" changes abruptly at the boundaries. Another example is "middle-aged". }

{ TO-DO: Prove that combinations of Γ and $\widetilde{\vee}$ $\widetilde{\wedge}$ can be universal approximators. }

4.3.13 $\mathcal{Z}$-conditionals

In general, inference is driven by rules. In $\mathcal{Z}$ logic all rules take the form of $\mathcal{Z}$-conditionals. A $\mathcal{Z}$-conditional is specified by a combination of min's and max's (similar to DNFs (disjunctive normal forms) in classical logic):

$$z_0 := \widetilde{\bigvee_i} \widetilde{\bigwedge_j} \Gamma(z_{ij}) \tag{4.5}$$

Notice that a $\mathcal{Z}$ rule directly assigns a $\mathcal{Z}$ value to z_0 *without the use of an implication operator*, which is very different from traditional fuzzy logics:

Traditional fuzzy logic

A fuzzy implication is a map $\Rightarrow$: $[0,1] \times [0,1] \to [0,1]$ satisfying these boundary conditions from binary logic:

$\Rightarrow$	0	1
0	1	1
1	0	1

A fuzzy implication statement: $(Z_1 \wedge Z_2) \Rightarrow Z_0$ means that the fuzzy values z_0, z_1, z_2 obey the equation:

$$((z_1 \wedge z_2) \Rightarrow z_0) = z_c$$

where z_c is the truth value of the implication statement. Compared to my approach, this has an extra level of indirectness. Is it really necessary that we know the truth value of an implication statement? (Cf §4.4.4: In probabilistic logic, the probability conditional $P(A|B)$ serves as the implication statement, but we usually

do not ask about its own probability.) One trouble with traditional fuzzy logic is that we cannot even perform *modus ponens* unless we allow interval fuzzy values.[7]

4.4 $\mathcal{P}$: probability

For an excellent introduction to probabilistic reasoning and Bayesian networks, see Judea Pearl's book [Pearl, 1988].

4.4.1 Why P is needed

In machine learning it is often necessary to learn facts that are only *contingently true*, such as the fact that "females often have long hair". Learning algorithms typically need to keep track of the frequencies of how often the hypotheses are true, in order to pick the highly probable ones. So it seems that probabilistic logic should be built into the knowledge representation.

4.4.2 Gaussian and Beta distributions

One very useful fact for designing AGI is that many quantities that occur naturally in our physical world seem to obey Gaussian distributions. This follows from the Central Limit Theorem which states that the sum of a large number of independent and identically-distributed random variables is approximately normally distributed.

The upshot of this is that the majority of fuzzy quantities, such as tallness, can be represented using Gaussian distributions. For example, the height of an unknown woman may have a Gaussian distribution with a mean of "5 feet 4". So we can just use 2 numbers, the mean and variance, to represent the distribution, instead of using a table which takes up more memory.

Since $\mathcal{Z}$ values are defined within $[0, 1]$, we can use the Beta distribution which is defined in the unit interval. It has parameters a, b:

$$f(x; a, b) = \frac{x^{a-1}(1-x)^{b-1}}{B(a, b)} \tag{4.6}$$

and is a very versatile and flexible distribution.

Sandy Zabell [Zabell, 1982] proved that, if we make certain assumptions about an individual's beliefs, then that individual must use the Beta density function to quantify any prior beliefs about a relative frequency [Neapolitan, 2004].

Some characteristics of its shape:
1. If $a > 1$ and $b > 1$ the shape is unimodal with the mode at $x = \frac{a-1}{a+b-2}$.
2. If $a < 1$ and $b < 1$ it is a U shape with an anti-mode at the same point, $x = \frac{a-1}{a+b-2}$.
3. If $a = b = 1$ it is the uniform distribution.
4. If $a = 1$ (respectively $b = 1$) then f(x) has a finite non-zero value at $x = 0$ (respectively at $x = 1$).
5. If $(a - 1)(b - 1) \leq 0$ it is J or reverse-J shaped, without a mode or anti-mode.
6. If $a = b$ the pdf becomes symmetric about $x = \frac{1}{2}$.
7. If $b > a$ the pdf is skewed to the right, and vice versa.

It is very useful that the mean μ and variance v are given by:

$$\mu = \frac{a}{a+b}$$

$$v = \frac{ab}{(a+b)^2(a+b+1)}$$

Further discussion about how the Beta distribution represents $\mathcal{P}(\mathcal{Z})$ variables is in §4.6.2.

[7] Suppose we define the operators for a very simple fuzzy logic: $a \Rightarrow b \equiv \neg a \vee b$, $\neg a \equiv 1 - a$, and $a \vee b \equiv min(a, b)$. [Kenevan and Neapolitan, 1992] has given an inference algorithm for this logic, but it is very complicated and involves interval fuzzy values, and so is not very suitable for further complex development.

4.4.3 First order logic and Bayesian networks

Early developments in Bayesian network are mainly propositional, which means that each node in a network represents a proposition without variables, such as "the fact that the alarm sounded". It has long been recognized that "lifting" Bayesian networks to first order is necessary for AI to be able to deal with open domains where *relations* can be defined over many *objects*.

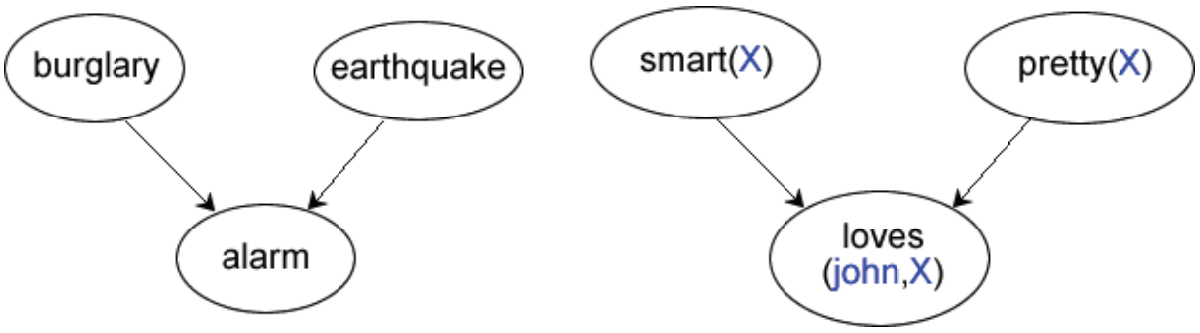


Figure 4.7: propositional vs first-order Bayesian networks

One key idea in first-order Bayesian networks is the method of Knowledge-Based Model Construction (KBMC), first developed by [Wellman, Breese, and Goldman, 1992] and [Haddawy, 1994]. The idea is to store a knowledgebase of first-order rules that can be used to construct propositional Bayesian networks on demand. When a query is asked, a Bayesian network is generated on-the-fly to answer the query. This idea helps us think of the first-order case in terms of the propositional case (though it may not be the most efficient implementation in practice).

Also, I have chosen the "directed" approach based on Bayesian networks, versus the "undirected" approach based on Markov networks (eg [Domingos and Richardson, 2007]'s Markov Logic Network (MLN)). It seems that the directed approach is more intuitive, in which probabilistic conditionals are used to represent *causal* relations.

See the book [Getoor and Taskar, 2007] for a collection of first-order probabilistic logic approaches.

Some first-order probabilistic logics:

[Kersting and de Raedt, 2000]'s Bayesian Logic Program (BLP)

[Laskey, 2006]'s Multi-Entity Bayesian Network (MEBN)

[Getoor, Friedman, Koller, Pfeffer, and Taskar, 2007]'s Probabilistic Relational Model (PRM)

[Milch, Marthi, Russell, Sontag, Ong, and Kolobov, 2007]'s Bayesian Logic (BLOG)

Other first-order probabilistic logics I have not looked into:

[Sato and Kameya, 1997]'s PRISM

[Muggleton, 1996]'s Stochastic Logic Program (SLP)

[Jaeger, 1997]'s Relational Bayesian Network (RBN), etc...

4.4.4 Bayesian networks and classical logic

Now we examine the relation between Bayesian networks and classical logic.

The problem with "material implication". The notion of implication in classical logic is no longer applicable in a probabilistic setting. To see why, consider the classical equivalence of implication

$$A \to B \equiv \neg A \vee B$$

and if we try to calculate the probability truth value of this statement we get

$$p = P(\neg A \vee B) = P(\neg A) + P(B) - P(B|\neg A)P(\neg A)$$
$$= 1 - P(A) + P(B|A)P(A)$$

using 3 basic rules of probabilities:

$P(X \vee Y) \equiv P(X) + P(Y) - P(X \wedge Y)$ (regardless of whether X,Y are independent)
$P(X \wedge Y) \equiv P(Y|X)P(X)$
$P(X) \equiv P(X|Y)P(Y) + P(X|\neg Y)P(\neg Y)$.

Now we have 4 things:

1. $P(A)$
2. $P(B)$
3. $P(B|A)$
4. $P(A \rightarrow B) = P(\neg A \vee B) = p$

but they cannot be fixed independently of each other: if we fix any 3 the 4th will also be fixed.

What is the problem here? The classical implication "$A \rightarrow B$" serves a function *similar* to the probabilistic conditional $P(B|A)$, but they constrain probabilities in slightly different ways, so they conflict with each other. If we keep both copies of #3 and #4 in our KB, and apply machine learning to learn their truth values, the values may fail to converge. It seems that classical implication is only *accidentally* equivalent to $\neg A \vee B$ when things are binary. In the probabilistic setting, we should jettison the binary implication in favor of the probabilistic conditional.

Adding probabilities to binary logic in a direct way (ie, using the binary material implication instead of $P(B|A)$) will result in very awkward inference algorithms (cf [Ng and Subrahmanian, 1992], [Nilsson, 1986]) that require setting up sets of inequalities for probability bounds. The result is so awkward that I think for all practical purposes this formulation can be said to be wrong. The Bayesian network formulation (using conditional probabilities $P(B|A)$) should be preferred.

Translating classical logic to Bayesian network. (Note: this translation is inexact but it preserves the intended meaning informally.) First, we can translate a first-order KB into Horn form. This is generally impossible, since Horn logic is a strict subset of first-order logic, but it can be done if we compile the knowledgebase into a pseudo-Horn form and use a special inference algorithm (this is done in [Stickel, 1988]). A Horn formula (which is equivalent to a Prolog statement) having the form:

```
A :- B, C, D, ...
```

would corresponds to:

$P(A|B, C, D, ...) = p$

in a Bayesian network. This approach is also adopted by BLP, MEBN, BLOG, and PRM (see §4.4.3).

4.4.5 Interval-valued probabilities

Sometimes, Bayesian networks fail to reproduce analogous results in classical logic unless we use interval probabilities. Consider this example:

> *China and the US are in conflict. If John sides with the US, he'll be a traitor. If he sides with China, he'll be a loser. Either way, John will be miserable.*

We can express the premises as conditional probabilities:

$P(\text{miserable} \,|\, \text{traitor}) = 0.9$
$P(\text{miserable} \,|\, \neg\text{traitor}) = 0.8$

and we want to query the probability $P(\text{miserable})$, but it is unknown whether John is a traitor or a patriot.

This example is exactly analogous to the "resolution rule" in classical logic. The classical inference step is:

traitor $\rightarrow$ miserable
$\neg$ traitor $\rightarrow$ miserable
———————————
miserable

If we construct a Bayesian network we will have the following CPT (conditional probability table):

traitor	**miserable**
true	0.9
false	0.8

but we cannot evaluate $P(\text{miserable})$ since $P(\text{traitor})$ is not known. This is a problem with point-valued Bayesian networks: *they fail to draw some analogous conclusions of classical logic*.

However, according to probability theory:

$$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B)$$

Therefore:

$$\begin{aligned} &P(\text{miserable}) = P(\text{miserable}\,|\,\text{traitor})P(\text{traitor}) + P(\text{miserable}\,|\,\neg\text{traitor})P(\neg\text{traitor}) \\ &= 0.9P(\text{traitor}) + 0.8P(\neg\text{traitor}) \\ &= 0.9p + 0.8(1-p) \\ &= 0.8 + 0.1p \\ &= [0.8, 0.9] \end{aligned}$$

In other words, if we allow the use of interval probability, we can infer that $P(\text{miserable}) = [0.8, 0.9]$ even when we assume that $P(\text{traitor}) = [0, 1]$ (ie, unknown). Thus we obtain a result analogous to classical resolution.

We need a Bayesian network inference algorithm that can handle this type of deduction, but first we consider an important simplification in the next section. The final algorithm will be given in §5.3.1.

4.4.6 Why point-valued probability is sufficient for AGI

In my opinion, second-order probability (such as interval probability or the indefinite probability developed by [Walley, 1991] and used in [Goertzel, Ikle, Goertzel, and Heljakka, 2008]) is an overkill for AGI. It makes the deduction algorithm very complex, and since the machine learning algorithm is based on deduction and is *even more* complex, the latter problem becomes practically impossible to solve in those settings. So we must make the deduction algorithm as simple as possible. Therefore I suggest using only point-valued probability.

In §4.4.5 I showed that interval probability is needed for some inference steps. That means the probability P itself is uncertain, and it lies in an interval. The *exact* algorithm for interval-probability inference requires us to set up the bounds of various probabilities and then invoke linear programming to solve for the probability bounds of the query variable (This method was first outlined by [Boole, 1854] and then developed by [Hailperin, 1965], [Nilsson, 1986], [Ng and Subrahmanian, 1992] et al. Recently [Hansen, Jaumard, de Aragao, Chauny, and Perron, 2000], [Jaumard and Parreira, 2006] developed faster algorithms for it, but still, the complexity of these algorithms is too much to handle if we want to design learning algorithms based on them.)

What I propose is that whenever we obtain an interval P value, we should convert it to a point value by *taking the mid-point of the interval*.[8]

For example, John may be unable to decide whether the president is smart or dumb. He may ascribe $P = [0.2, 0.8]$ to the atom $smart(president)$. According to my scheme, he can use

$$P = (0.2 + 0.8)/2 = 0.5$$

as a compromise. This is like saying "there's 50-50 chance". Is this approximation too bad? It seems that many people think like this anyway. I'd be surprised if the human brain maintains 2nd-order probabilities internally.

Moreover, the exact values of P often do not affect our behavior that much. There is evidence that taking the centroid (the center of mass of a belief distribution) can yield reasonably good results in second-order probabilistic decision-making ([Sundgren, Danielson, and Ekenberg, 2006]). Also, [Bier, 1993] shows that there are broad classes of utility functions for which uncertainty is irrelevant under expected utility theory, and only mean values are significant. { TO-DO: There is hand-waving here. }

Maybe in a much more advanced AGI, we would want 2nd-order precision, but that seems not to be the right priority now. It may be more effective for an AGI to improve its decisions by:

1. considering more factors;
2. updating probabilities using more evidence;
3. refining explanations (causal relations); etc.

Using point-valued probabilities (without knowing their error) is not such a big sin, if we compare this with what we do in fuzzy logic all the time. A fuzzy statement such as:

[8] We still need inference algorithms that can handle intervals as demonstrated in §4.4.5, but the intervals will be instantly converted to point-values after each step. This reduces complexity greatly.

"John is fairly tall"
is often represented simply by:
tall(john) $z = 0.7$
which is analogous to representing the probabilistic statement
"John is usually punctual"
with a point-valued probability:
punctual(john) $p = 0.8$.

If fuzziness is a more fundamental phenomenon in our knowledge representation, we should be making more fuss about fuzziness than probabilities. It seems that we ascribe more "prestige" to probability theory merely because of psychological reasons.

4.4.7 Probabilistic AND and OR

Specifying the CPT (conditional probability table) of a single node of a Bayesian network, if the node has n parents, would require 2^n entries. The "noisy" AND and OR gates are designed to simplify this by reducing the number of independent entries to n. The interpretation of the noisy OR gate is that each parent variable X is associated with an "inhibition probability", q ([Pearl, 1988] p184-187 or [Russell and Norvig, 2003] p500-501). This is the textbook definition of noisy OR (and I created noisy AND by applying DeMorgan's laws[9]):

		noisy AND	**noisy OR**
X_1	X_2	$X_1 \curlywedge X_2$	$X_1 \curlyvee X_2$
0	0	$1?$	0
0	1	$1 - q_1?$	$1 - q_1$
1	0	$1 - q_2?$	$1 - q_2$
1	1	$1 - q_1 - q_2 + q_1 q_2?$	$1 - q_1 q_2$

It seem that this definition of noisy AND is problematic (for example, the "1" there should be close to 0).

{ TO-DO: Abram Demski pointed out that the textbook definition of noisy-OR is actually OK and there is no need to invent a new one. }

So, I define my version of "probabilistic" AND and OR via DeMorgan's laws. Each variable is attached with a "causal strength" $c \in [0, 1]$, such that when $c \to 1$ they reduce to classical AND and OR. (But the original "noisy-OR" interpretation is lost.)

		probabilistic AND	**probabilistic OR**
X_1	X_2	$X_1 \curlywedge X_2$	$X_1 \curlyvee X_2$
0	0	$(1 - c_1)(1 - c_2)$	$1 - c_1 c_2$
0	1	$(1 - c_1)\ c_2$	$1 - c_1 + c_1 c_2$
1	0	$c_1\ (1 - c_2)$	$1 - c_2 + c_1 c_2$
1	1	$c_1\ c_2$	$c_1 + c_2 - c_1 c_2$

(4.7)

This can be easily generalized to $n > 2$.

A CPT can be defined by a combination of probabilistic AND-OR's:

$$X_0 := \bigvee_i \bigwedge_j X_{ij} \tag{4.8}$$

where the X_{ij}'s are parents of the node X_0 in the Bayesian network.

Notice that in the above equation, each connective is associated with a pair of c parameters, so the actual equation is:

$$X_0 := \bigvee_i \bigwedge_j X_{ij}; c_{ij} \ = \bigvee \{X_{11}; c_{11} \curlywedge X_{12}; c_{12} \curlywedge \cdots, X_{21}; c_{21} \curlywedge X_{22}; c_{22} \curlywedge \cdots, \cdots\} \tag{4.9}$$

And, because of the c parameters, association does not hold in general, ie:
$((A \curlyvee B) \curlyvee C) \neq (A \curlyvee (B \curlyvee C))$.

[9] That is, to require $\overline{X_1 \curlyvee X_2} \equiv \overline{X_1} \curlywedge \overline{X_2}$ and $\overline{X_1 \curlywedge X_2} \equiv \overline{X_1} \curlyvee \overline{X_2}$

4.5 $\mathcal{C}$: confidence

Pei Wang's uncertain logic is particularly elegant. I adopt two ideas from his theory, explained below, but the way I use those numbers differs slightly from Wang's (cf his book [Wang, 2006] which describes an AGI called NARS (Non-axiomatic Reasoning System)).

4.5.1 Positive and negative evidence

In Wang's logic, each statement is attached with 2 numbers:

w^+ = number of positive examples

w^- = number of negative examples

In an AGI system, they are the number of times a statement is observed to be true or false, respectively.

For example, for the statement

"if X is female X probably has long hair"

the AGI may have observed 70 females with long hair and 30 females with short hair. The total **support** for the statement would be $(w^+ + w^-) = 100$ examples.

Using a pair of numbers allows us to know the probability of a statement as well as the the number of examples that support that statement. This is very important because some statements may be supported by very few examples and thus are "weaker" than statements with more support.

A major advantage of this 2-number approach is that probabilities can be updated by new examples. For example, if the AGI encounters a new female with long hair, it can update the probability easily with $(w^+ + 1,\ w^-)$. Such updating cannot be done with the single-number representation of probability.

4.5.2 Confidence

Confidence is a concept borrowed from NARS. In my terminology, the **support** of a statement is defined as:

$$w = w^+ + w^- \tag{4.10}$$

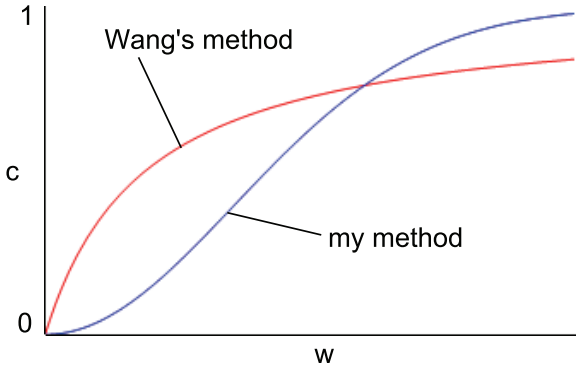
which is an integer from 0 to ∞.

As Pei Wang did in NARS, the confidence c is the value obtained by squashing w into $[0, 1]$, using a nonlinear transform such as:

$$c = \frac{1}{1 + k/w} \qquad \text{(Wang's method)} \tag{4.11}$$

or

$$c = 1 - e^{-ln2 \cdot (w/k)^2} \qquad \text{(my method)} \tag{4.12}$$



An advantage of Wang's method is that it allows statements to have substantial confidence *earlier* (so it gives nascent hypotheses more chance to succeed). The advantage of my method is that it allows confidence to get closer to 1 sooner (so the incumbents have more strength). The choice between these 2 may have interesting consequences in machine learning (see §5.3.3).

4.5.3 Confidence and probability

The probability (or frequency) of a statement is simply:

$$p = f = \frac{w^+}{w} \tag{4.13}$$

Thus, (w^+, w^-) contains information equivalent to the frequency and confidence pair (f, c).

Notice that the confidence c is *orthogonal* to f or p. In §4.6.1 we will see that each truth value in our logic is represented as a tuple: (μ, v, c) where (μ, v) describes a probability distribution and c is the confidence.

4.5.4 Confidence and logic

To see how confidence is defined on logical formulae, it is easier to think in terms of the support w. The support of a **rule** (ie, a logical formula containing variables) can be defined in a frequentist manner; whereas the support of a **ground fact** (ie, a formula without variables) is always *derived* from inference.

For example, the rule:

"Women usually have long hair"

female(X) $\rightarrow$ long-hair(X)

is supported by a number of instances such as:

positive example: female(mary), long-hair(mary)

negative example: female(jane), $\neg$ long-hair(jane)

The support of the rule is the *total* number of such instances that have been encountered[10].

A ground fact cannot be given the same treatment because it has no instances (being itself an instance) and its confidence must be inferred. Inference of confidence is treated in §5.3.3.

4.6 Combining $\mathcal{B}$, $\mathcal{P}$, $\mathcal{Z}$

4.6.1 The truth value P(Z)

How to combine $\mathcal{P}$ and $\mathcal{Z}$? The answer is simple because there is no other choice: the semantics of probabilities dictate that $\mathcal{P}$ must be *distributed over events*. In the current system, events are either $\mathcal{B}$ or $\mathcal{Z}$ (the latter are *continuous* events). So we *distribute* $\mathcal{P}$ *over* $\mathcal{B}$ *and* $\mathcal{Z}$. In this sense, fuzziness is more fundamental than probabilities.

If a $\mathcal{Z}$ value is uncertain — for example, we may not be sure how tall Mary is (the $\mathcal{Z}$-value of her tallness may be 0.6-0.8, say, so we can assume a uniform probability distribution over the interval [0.6,0.8] which is the green rectangle below) — and we can approximate it by a Beta distribution over $\mathcal{Z}$ with a mean at 0.7 and the same variance:

[10] Here the Raven's paradox (or Hempel's paradox) is relevant: Given the rule that "women usually have long hair" we can also state conversely that "people with short hair are usually not women". Thus, a man with short hair would become a supportive example of this rule, which is counter-intuitive. { TO-DO: resolve this }

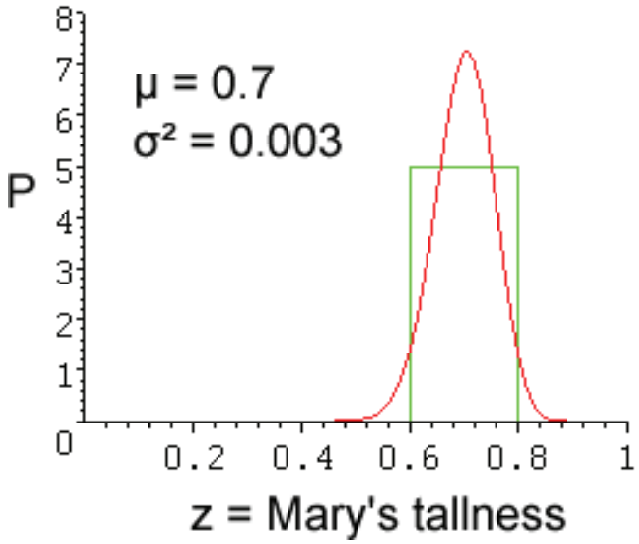


Figure 4.8: an example $\mathcal{P}(\mathcal{Z})$ distribution

where the probability density should sum to 1: $\int_0^1 P(z)dz = 1$.

On the other hand, if a $\mathcal{P}$ value is uncertain, we simply *ignore* its error (eg, by choosing the mid-point of a P-interval). §4.4.6 tried to justify this.

4.6.2 Unifying all truth values to $\mathcal{P}(\mathcal{Z})$

Up to now there are 4 possible TV types:

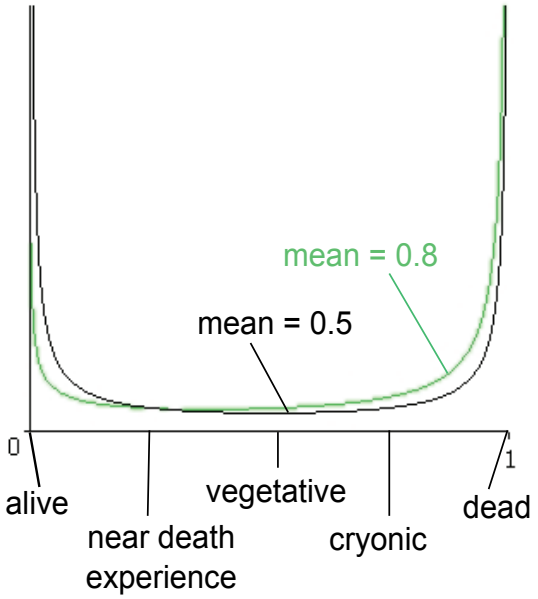
Table 4.2:

truth values		
mnemonic	meaning	definition
$\mathcal{B}$	binary	$b \in \{true, false\}$
$\mathcal{Z}$	fuzzy	$z \in [0, 1]$
$\mathcal{P}(\mathcal{B})$	$\mathcal{P}$ distributed over $\mathcal{Z}$	$P(b = false) = p_0$ $P(b = true) = p_1$
$\mathcal{P}(\mathcal{Z})$	$\mathcal{P}$ distributed over $\mathcal{Z}$	$P(z = z_1) \sim Beta(z_1)$

I find that they can be unified to type $\mathcal{P}(\mathcal{Z})$, which can make things simpler. Below is how to represent the other 3 TV types as $\mathcal{P}(\mathcal{Z})$:

Type $\mathcal{B}$

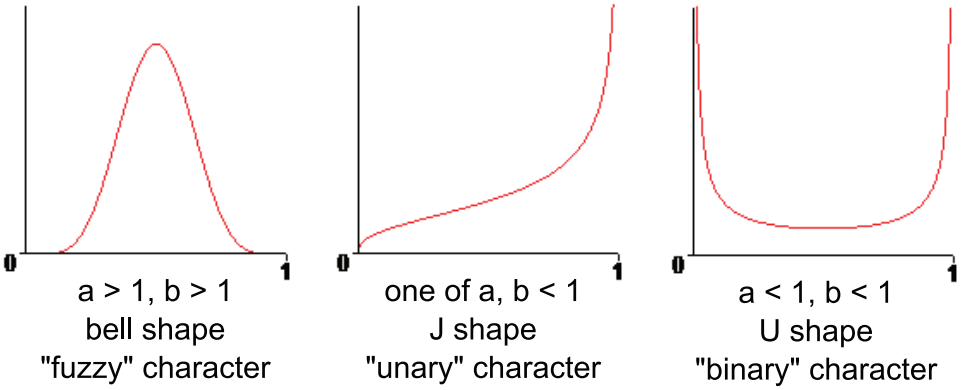
Some variables have a strong "binary flavor". For example, in common sense, a person is either dead or alive, although a more nuanced view will have grades of being dead. If deadness is a $\mathcal{Z}$ variable, z = 0 would be "definitely alive", z = 1 would be "definitely dead" (eg reduced to ash), and z = 0.5 would correspond to a state that is difficult to classify as dead or alive, eg a brain-dead, vegetative state. z = 0.7 may be a state that is more dead than brain-dead and yet more alive than ash, eg — I have to pause for a while to think of an example — a body under cryonic preservation. And z = 0.4 may be some kind of near-death experience. Anyway, one can expect the probability distribution of z_{dead} to be polarized with a trough in the middle. This can be represented by a Beta distribution with a large variance:



When the polarization gets extreme (eg the toss of a coin is either head or tail), the distribution becomes a ⊔ shape. The degree of polarization (ie amount of variance) can be estimated statistically, if data is available.

It appears that all common-sensical $\mathcal{B}$ variables are actually **polarized** $\mathcal{Z}$ variables. Another example is a person being either married or not married, and there are grey areas like gay marriage or marriage for the green card, etc.

Actually the Beta distribution (eqn 4.6) is capable of representing 3 types of characteristics (with $a = 1$ and $b = 1$ as points of transition separating the 3 regimes):



The unary type represents concepts such as "normal", "broken", "healthy", or "natural". In such concepts, the state at $z = 1$ can be clearly defined (eg, John was perfectly healthy when he was a kid), but as we approach $z = 0$ the cases become very improbable and inexhaustible (eg, it is impossible to find a person who is "utmost unhealthy" because it is always possible to think of more extreme and improbable unhealthy ways. Also, committing suicide is not the limiting case — suicide is not the same as unhealthy — so the interval is open-ended.)

About the mean and variance. In the fuzzy regime, the smaller the variance, the shaper the peak and thus the more confident we are about the $\mathcal{Z}$ value; The variance is a measure of confidence in this regime. In the binary regime, the greater the variance, the more *polarized* the distribution becomes; The variance is a measure of binary character. In the unary regime, the meaning of the variance is unclear.

To illustrate this with an example: In the fuzzy regime, I can say "John is 0.7 dead because he's in a cryonic state", and I can use a small variance to indicate that I am confidence that John is in a cryonic state and *not elsewhere*. In the binary regime, however, I can only indicate the probability of "John being dead or not" where "dead" is an "either-or" condition. The highest probabilities are concentrated at 1 ("definitely dead") and 0 ("definitely alive"), and the cryonic state has low probability. Under this regime there is no way to accentuate (make more probable) the cryonic state — one can only do so in the fuzzy regime. This is exactly what we would expect with a binary variable.

Type $\mathcal{Z}$

We can create a $\mathcal{P}(\mathcal{Z})$ distribution with a peak around z. The variance depends on how confident we are of the z value. If unspecified, we can assign a typical variance to it.

Type $\mathcal{P}(\mathcal{B})$

For example, "Mary is probably married", with p = 0.8.
ie $P(married) = 0.8, P(\neg married) = 0.2$.

To convert this into a $\mathcal{P}(\mathcal{Z})$ distribution, one can set the probability mass for $z \geq \frac{1}{2}$ to be equal to p. This can conveniently be done via the CDF of the Beta distribution, which is the regularized incomplete Beta function:

$$F(x; a, b) = \frac{B(x; a, b)}{B(a, b)} = I_x(x; a, b)$$

where $B(x)$ is the incomplete Beta function:

$$B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1}dt$$

The desired mean value is given by

$$I_x(\mu; a, b) = p \tag{4.14}$$

The variance represents the degree of polarization of the variable, which varies from one variable to another, and can be quite arbitrary if unspecified.

Inference of $\mathcal{P}(\mathcal{Z})$ logic will be treated in §5.

4.7 Meta-reasoning

Meta-reasoning is an important topic that will be treated in §8.

One example of meta-reasoning pertinent to fuzziness is:

S1: "You are either a patriot or a traitor."

S2: "No, I can be slightly patriotic or slightly traitorous."

In S1 the predicates *patriot* and *traitor* have binary character. In S2 they have fuzzy character.

5 Inference

A syllogism has 3 parts; therefore, this is not a syllogism.

By inference we mean deduction and abduction; induction is regarded a form of learning.

What are the computational bottlenecks? How to speed things up?

5.1 Some background

5.1.1 Resolution

5.1.2 Horn clauses and Prolog

5.1.3 Forward- and backward- chaining

5.1.4 Complexity of inference

Some facts:

1. Satisfiability in binary **first-order logic** is semi-decidable (ie, it terminates in finite time if a proof exists, but may never terminate if a proof does not exist)

2. SAT for binary **propositional logic** is NP-complete.

3. **Propositional Bayesian network** inference is #P-complete (ie, it is as hard as returning the number of solutions to an NP-complete problem).

4. The *naive* algorithm for propositional Bayesian network inference is $O(n2^n)$ where n is the number of nodes.

5. Pearl's message-passing algorithm is an *exact* algorithm for propositional Bayesian network inference; therefore it also requires non-polynomial time in the worst case.

6. Pearl's algorithm, when operating on trees, is linear time. But trees seem to be insufficient for AGI reasoning.

7. SAT for the **propositional Horn** subclass of binary logic, can be performed in linear time. My very simple algorithm for Bayesian inference is analogous to Horn deduction[1]. Prolog owes its efficiency to SLD resolution which is also Horn-based.

5.2 Nonmonotonicity and redundancy

5.2.1 Handling exceptions (nonmonotonic reasoning)

Fuzzy reasoning can sometimes lead to unsound conclusions, for example:
1. *Mary has cybersex with many partners.*
2. *Cybersex is a kind of sex.*
3. *Therefore, Mary has many sex partners.*
4. *A person who has many sex partners has a high chance of STDs.*
5. *Therefore, Mary has a high chance of STDs.*

What's wrong with this example? On the one hand, we should admit that cybersex is sex (it is a borderline case), but it lacks certain prominent features of sex, such as physical contact (which is not necessarily a *defining* feature of sex). Thus, if we carry on reasoning with the idea that cybersex is sex, we may get unsound conclusions. The key to resolving this problem is to recognize "cybersex is sex" with **qualifications** such as "but it is sex without physical contact".

If we have a rule saying that "sex transmits certain diseases", we may have to attach the exception "only if the sex involves physical contact". In the end, our rules may be inundated with possibly infinitely many exceptions. How can we get out of this problem?

[Wang, 1994], [Wang, 2006] provided a solution, which I essentially adopt. His idea is not to store the exceptions to rules, but instead allow a *multitude* of rules to fire, calculate the "confidence" (§4.5) of each conclusion, and pick the conclusion with the highest confidence. This allows us to handle exceptions relatively easily.

5.3 Deduction

5.3.1 $\mathcal{P}$ inference algorithm

TO-DO: Deal with loops.

TO-DO: There should be a mechanism to prune low-probability (or should it be low-confidence?) branches early on in the search.

TO-DO: Compare $\mathcal{P}$ inference with standard Bayesian network inference algorithms...?
This algorithm merely outlines the rules involved in computing $P(X)$.

Problems of Bayesian network inference:
1. Some inference steps analogous to classical logic cannot be performed because probabilities are not truth-functional (ie, the probability of a statement is not the function of the probabilities of its constituents; it may depend on other probabilities not appearing in the statement).
2. Bayesian network inference cannot be performed one-rule-per-step because of interdependence of variables that requires message passing.

Currently I am using an extremely simplified approximation.

5.3.2 Hybrid $\mathcal{B}, \mathcal{P}, \mathcal{Z}$ inference

Rules, which are the building blocks of inference, can be of 4 types:

[1] The term Horn can only be used to describe binary logic: a Horn clause is a clause with exactly one positive literal. $\mathcal{PZ}$ logic has a Horn-like form, but the distinction between positive and negative literals disappears in logics with numerical truth values.

Table 5.1:

rules	
mnemonic	meaning
$\mathcal{B}$	Prolog statements of the form $L_0 :\!- L_1, L_2, L_3, ...$
$\mathcal{Z}$	Fuzzy rules of the form $z_0 := \widetilde{\bigvee} \widetilde{\bigwedge} z_{ij}$
$\mathcal{P}(\mathcal{B})$	Bayesian network nodes specified by $X_0 := \curlyvee \curlywedge X_{ij}; c_{ij}$
$\mathcal{P}(\mathcal{Z})$	Specifications of $\mathcal{P}(\mathcal{Z})$ distributions $X_0 := Beta(\mu, \sigma^2)$

In principle, a unified form of $\mathcal{P}(\mathcal{Z})$ rule is possible, but I have found it to be too complex and unwieldy to use. So we will use a hybrid mixture of the above 4 types of rules.

The type of a rule is determined by its "head" — eg, if the head is a $\mathcal{B}$ variable then the rule is a $\mathcal{B}$ rule. Using unified $\mathcal{P}(\mathcal{Z})$ truth values, we can plug variables of any TV type into any rule. So, the rule type only affects how the rule is interpreted, which will be explained below.

An example of a rule is: "almost Q" means "close to Q, but not being Q".
The TV types of the predicates are:
"almost Q" — $\mathcal{B}$
"close to Q" — $\mathcal{Z}$
"not Q" — $\mathcal{B}$
So we can represent it with a $\mathcal{B}$ rule:
almost $Q \leftarrow$ close-to $Q \wedge \neg Q$ Despite using a $\mathcal{B}$ rule, the result of inference will be a $\mathcal{P}(\mathcal{Z})$ truth value with binary character. This is a good thing, because the idea of "almost" can be fuzzy in some cases, for example:
"I heard that John almost got killed by the assassination?"
"Not really, the bullet only hit his toe!"

To simplify things, I consider single inference steps. A complete proof involves a series of such steps.

$\mathcal{B}$ rule:

The $\mathcal{B}$ rule can have 3 operators: $\wedge$, $\vee$, and $\neg$. All we need to do is to show how to evaluate the operators for $\mathcal{P}(\mathcal{Z})$ values. An example $\mathcal{B}$ rule is:
`bachelor :- male` $\wedge \neg$ `married`

Case $\neg$: We need to convert the $\mathcal{P}(\mathcal{Z})$ distribution to one with a binary character. This is done by setting the variance to a fixed value in the binary regime (§4.6.2). Then the result is negated by transforming the mean:

$$\mu' = 1 - \mu \tag{5.1}$$

Case $\wedge$: Convert the $\mathcal{P}(\mathcal{Z})$ value to $\mathcal{P}(\mathcal{B})$ via eqn (4.14). Then $P(A \wedge B) = P(A)P(B)$, assuming independence.

The variance is assigned a fixed value so that the resulting variable has binary character. Note that the resulting variance is not affected by the variances of the premises because the aim of a $\mathcal{B}$ rule is to form a *binary* judgment.

Case $\vee$: Similar to above, with $P(A \vee B) = P(A) + P(B) - P(A)P(B)$, again assuming independence.

$\mathcal{Z}$ rule:

$\mathcal{Z}$ rules can have the operators $\widetilde{\wedge}$ (= min), $\widetilde{\vee}$ (= max), $\widetilde{\neg}$ (= $1 - z$) and the fuzzy modifier $\Gamma()$. At this stage we ignore Soft min and max. The $\Gamma()$'s are applied before the other operators.

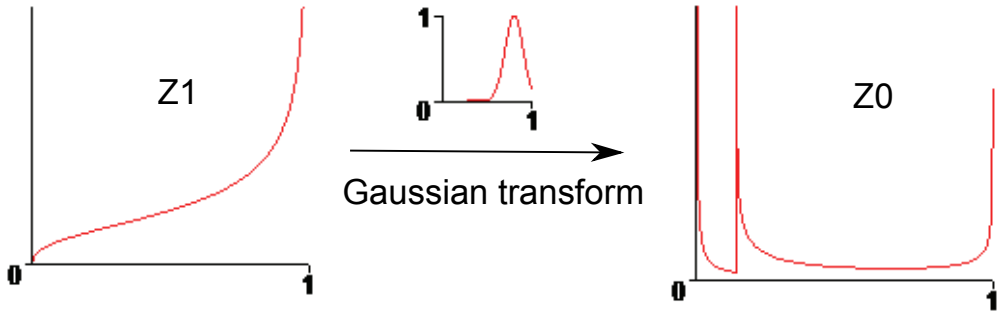
Case $\Gamma()$: If $z_0 := \Gamma(z_1)$ and z_1 is given by the probability density function $f_1(z1)$, then the probability density of z_0 would be given by:

$$f_0(z_0) = f_1(\Gamma^{-1}(z_0)) \left| \frac{d\Gamma^{-1}}{dz_0} \right| \tag{5.2}$$

which is explained in [Wikipedia, 2008]. If $\Gamma()$ is the Gaussian function given in eqn (4.4) then

$$f_0(z_0) = f_1(\pm\sigma\sqrt{-2\ln z_0} + z^*)\left|\frac{\sigma}{z_0\sqrt{-2\ln z_0}}\right|$$

but there is a glitch: $\Gamma^{-1}()$ has 2 pieces, and their contributions must be summed up piecewise. Sometimes the resulting distribution looks irregular, for example:



but we can approximate it with a Beta distribution by preserving the mean and variance. The irregular appearance does not matter that much if we only care about the mean and variance. I have obtained the approximation formula by numerical integration and nonlinear regression on randomly generated samples, as follows (m_1 = mean of input, m_0 = mean of result):

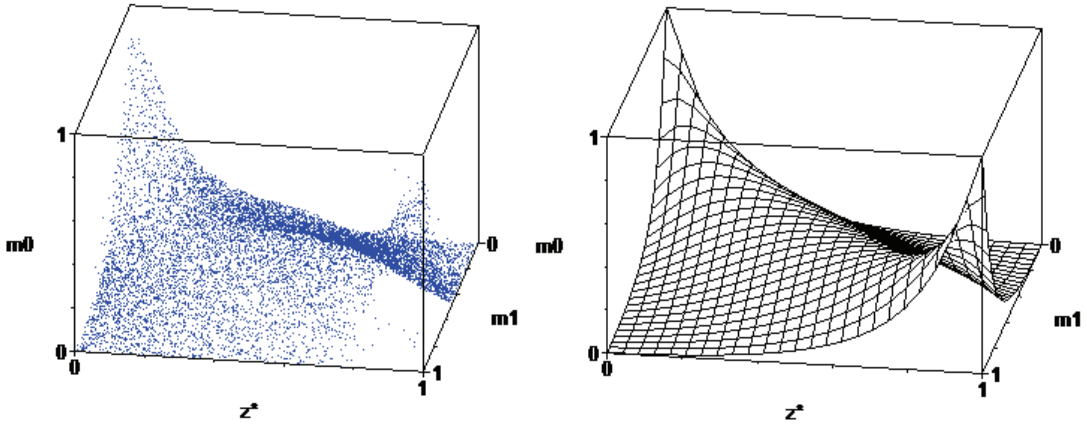


Figure 5.1: Gaussian fuzzy modifier: empirical data and result of regression

Case $\neg$: Simply negate the $\mathcal{P}(\mathcal{Z})$ distribution by eqn (5.1) with variance unchanged.

Case $\widetilde{\vee}$: Given $z_0 := z_1 \widetilde{\vee} z_2$, ie $z_0 = max(z_1, z_2)$. What we need to do here is to calculate the PDF of a random variable given as a function of two other random variables. The procedure is given in many textbooks.

Here $\mathbf{P}$ denotes probability measure which is a set function, $F(t)$ denotes CDF's, $f(t)$ denotes PDF's.

$$\begin{aligned}
F_0(t) &= \mathbf{P}(Z_0 \le t) \\
&= \mathbf{P}(\widetilde{\vee}(Z_1, Z_2) \le t) \\
&= \mathbf{P}(max(Z_1, Z_2) \le t) \\
&= \mathbf{P}(Z_1 \le t \wedge Z_2 \le t) \\
&= \mathbf{P}(Z_1 \le t)\,\mathbf{P}(Z_2 \le t) \quad \text{assuming } Z_1, Z_2 \text{ independent} \\
&= F_1(t)F_2(t)
\end{aligned}$$

where

$$F_1(t) = \int_0^t f_1(z_1)dz_1, \quad F_2(t) = \int_0^t f_2(z_2)dz_2$$

The result we want is:

$$f_0(t) = \frac{dF_0(t)}{dt} = F_2(t)\frac{dF_1(t)}{dt} + F_1(t)\frac{dF_2(t)}{dt}$$

and we need to apply Leibnitz's Rule.[2] Then we get:

$$f_0(t) = f_1(t)F_2(t) + f_2(t)F_1(t) \tag{5.3}$$

This function f_0 has an interesting property: it appears to add up the distributions f_1 and f_2, *with the one on the right being dominant*, ie, giving the biggest contribution of probability mass (or area under the curve). Some examples are as follows:

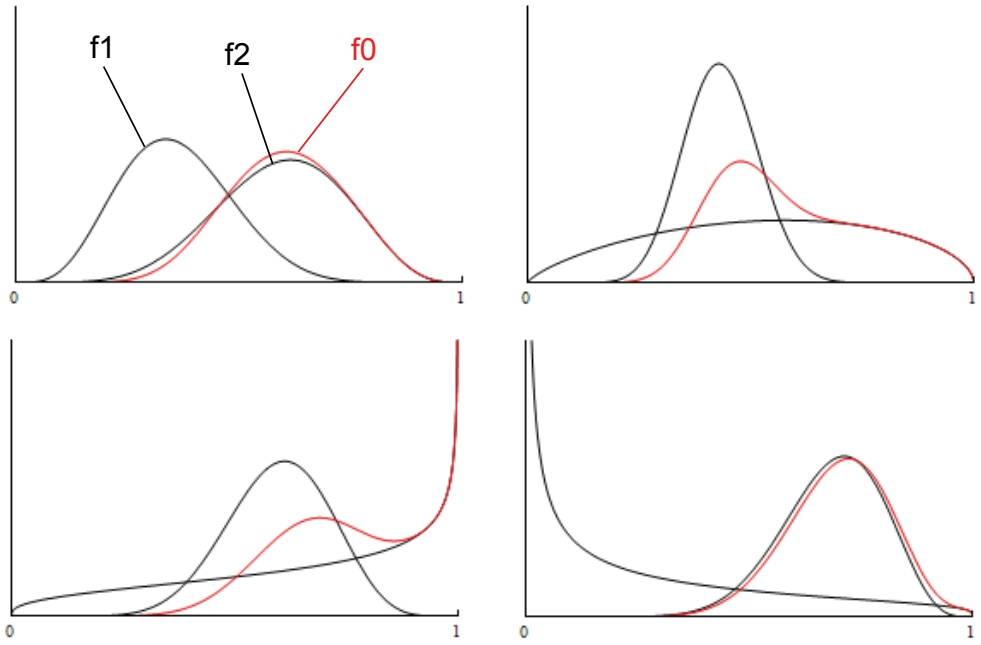


Figure 5.2: $z_0 := z_1 \,\widetilde{\vee}\, z_2$

Once again, we approximate by preserving the mean and variance; The irregular appearance is not so important. By looking at the following graph we can see that the mean of the result (m_0) is mostly dominated by m_2 which is the mean of the input further to the right (ie, $m_2 > m_1$). There is sometimes a slight shift to the right relative to m_2 (the dots above the $x = y$ line), but it seems insignificant and may be ignored. In other words, the fuzzy inference rule is very simple: *the mean of the result is just the greater mean of the 2 inputs*. A similar graph shows that the variance of the result is also approximately equal to the variance of the rightmost input.

[2] For a function of t defined by:

$$F(t) := \int_a (t)^b(t)\Phi(x,t)dx$$

its differentiation is given by Leibnitz's Rule:

$$\frac{dF(t)}{dt} = \int_{a(t)}^{b(t)} \frac{\partial\Phi(x,t)}{\partial t}dx + \Phi(b(t),t)\frac{db(t)}{dt} - \Phi(a(t),t)\frac{da(t)}{dt}$$

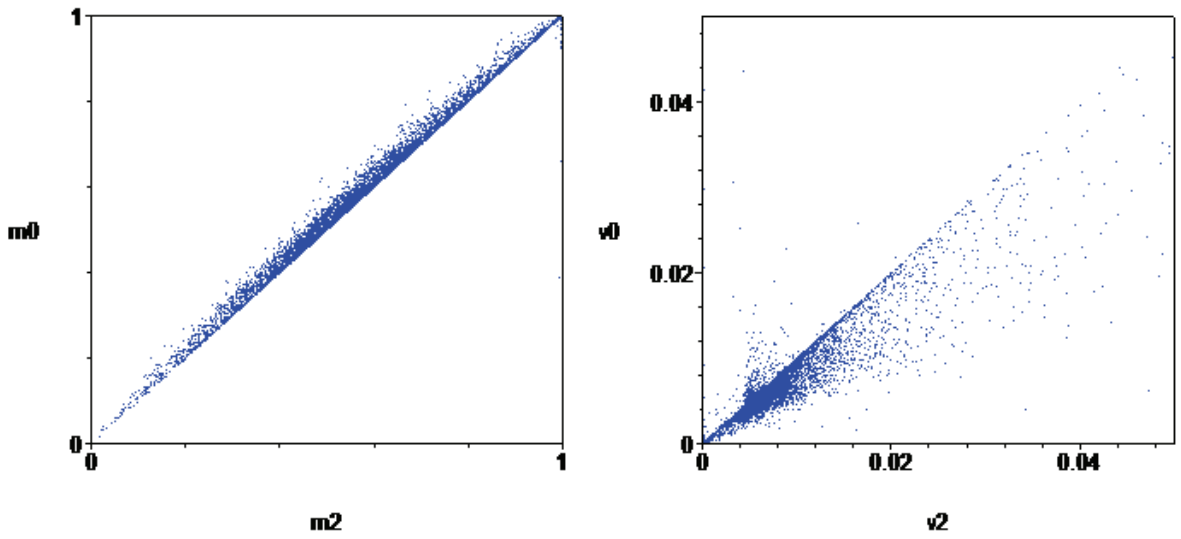


Figure 5.3: Plots of mean and variance; output vs input

Case $\widetilde{\wedge}$: The result for $z_0 := z_1 \,\widetilde{\wedge}\, z_2$ is similar (again assuming Z_1, Z_2 independent):

$$f_0(t) = f_1(t) + f_2(t) - f_1(t)F_2(t) + f_2(t)F_1(t) \tag{5.4}$$

In summary, the fuzzy inference rules are:

1. For $z_0 := \Gamma(z_1)$,

$$\mu_0 = \frac{k_1 + k_2}{\sigma_1} e^{-(m_1 - z^*)^2/\sigma_1^2}$$

$$v_0 = \frac{k_3}{\sigma_2} e^{-(z^* - \sqrt{3}m_1 + k_4)^2/\sigma_2^2}$$

   where $\sigma_1 = k_1(m_1 + z^* - 1)^2 + k_2$
   $\sigma_2 = k_5(z^* + \sqrt{3}m_1 - k_6)^2 + k_7$
   and the k's are regression parameters
2. For $z_0 := \neg z_1$, $\mu_0 = 1 - \mu_1$, $v_0 = v_1$
3. For $z_0 := z_1 \widetilde{\vee} z_2$ or $z_1 \widetilde{\wedge} z_2$,
   $\mu_0 = max(\mu_1, \mu_2)$ or $min(\mu_1, \mu_2)$,
   $v_0 = v_1$ or v_2 following the choice above

$\mathcal{P}(\mathcal{B})$ rule:

A $\mathcal{P}(\mathcal{B})$ rule is of the form:

$$X_0 := \bigvee \bigwedge X_{ij}; c_{ij}$$

To simplify things, we do not implement the Bayesian network algorithm, instead we only perform the calculation X_0 from X_{ij} in one direction. This is an extreme simplification of probability logic, but it may already be sufficient for common-sense reasoning. We will try this first and see how far it can go.

The $\mathcal{P}(\mathcal{Z})$ variables can be converted to $\mathcal{P}(\mathcal{B})$ variables via eqn (4.14). Then the $\mathcal{P}(\mathcal{B})$ value of the consequent can be obtained via straightforward application of probabilities, eg:

$$\begin{aligned} P(X_1 \wedge X_2) = {} & (1-p_1)(1-p_2)(1-c_1)(1-c_2) + \\ & (1-p_1)p_2(1-c_1)c_2 + \\ & p_1(1-p_2)c_1(1-c_2) + \\ & p_1 p_2 c_1 c_2 \end{aligned}$$

where $p_i = P(X_i)$. Note that it reduces to the binary case when $c_1 = c_2 = 1$. The result is then converted back as a $\mathcal{P}(\mathcal{Z})$ distribution (with binary character).

$\mathcal{P}(\mathcal{Z})$ rule:

We do not use truly $\mathcal{P}(\mathcal{Z})$ rules because they are too complex and not needed for common-sense reasoning. The $\mathcal{P}(\mathcal{Z})$ rules we have are simpler operations that allow us to manipulate the $\mathcal{P}(\mathcal{Z})$ distributions of variables.

We can directly assign the mean and variance to a $\mathcal{P}(\mathcal{Z})$ variable:

$$X_0 := Beta(z; a, b)$$

Or we can fix the probability at a particular point $z*$'s neighborhood. For example, "the probability of Mary being 0.9 fat is 0.1".

5.3.3 Inference of $\mathcal{C}$

Confidence was introduced in §4.5. There, I explained that:
1. The confidence of rules can be measured in a frequentist way[3]
2. The confidence of ground facts must be inferred

The problem is, we must initially know the confidence of at least *some* ground facts in order to infer those of others. To this end, I propose a convention to assign all **raw sensory events** (eg, "a camera's pixel records the RGB color #556677 at time 012345") to have confidence 1 (and thus ∞ support). (It should not have 0 confidence because that would mean, informally, that the statement is so weak as to have no support at all).

In general, each inference step involves a rule of the form:

A ⟵ B, C, D, ...

where the rule has a frequentist confidence c_R, and the antecedents B, C, D,... each has an inferred confidence. So how to calculate c_A from c_B, c_C, c_D, ...?

Let's start from the simplest case. Consider the rule:

A ←———— B
? c_R c_1

where c_R = confidence of the rule
c_1 = confidence of the antecedent B
and we seek the formula for $c_0 := f(c_1, c_R)$.

By considering the following "boundary" cases:
(the numbers are confidences)

A ←———— B
(0) 0 1
A ←———— B
(0) 1 0
A ←———— B
(0) 0 0
A ←———— B
(1) 1 1

we surmise that $f(\cdot, \cdot)$ degenerates into binary AND. In other words, we're seeking an operator that generalizes binary AND. Naturally, this can be either probabilistic AND (the product rule) or fuzzy AND (min rule).

The min rule epitomizes the proverb "a chain is only as strong as its weakest link". Its special property is that a rule of < 1 confidence can be repeatedly applied without decreasing the conclusion's confidence.

The probabilistic rule forces confidence to *decay* in long inference chains, if the confidence of each step is < 1.

At this point it is hard to say which method is superior.

To recap: Every rule has a frequentist confidence; a ground fact's confidence is inherited from the rules that entail it.

Confidence is useful in 2 ways:
1. As a termination criterion for proof-search: A statement of low probability can still be significant, eg: "it is highly unlikely that an AGI can be run on an APPLE][". A statement is insignificant if its confidence is

[3]Note, however, that the confidence c is not a frequency; it is just related to the frequency.

close to 0.
2. In belief revision, when 2 inference chains arrive at different conclusions, their confidences will decide which has the greater weight.

5.3.4 Putting it all together: the deduction algorithm

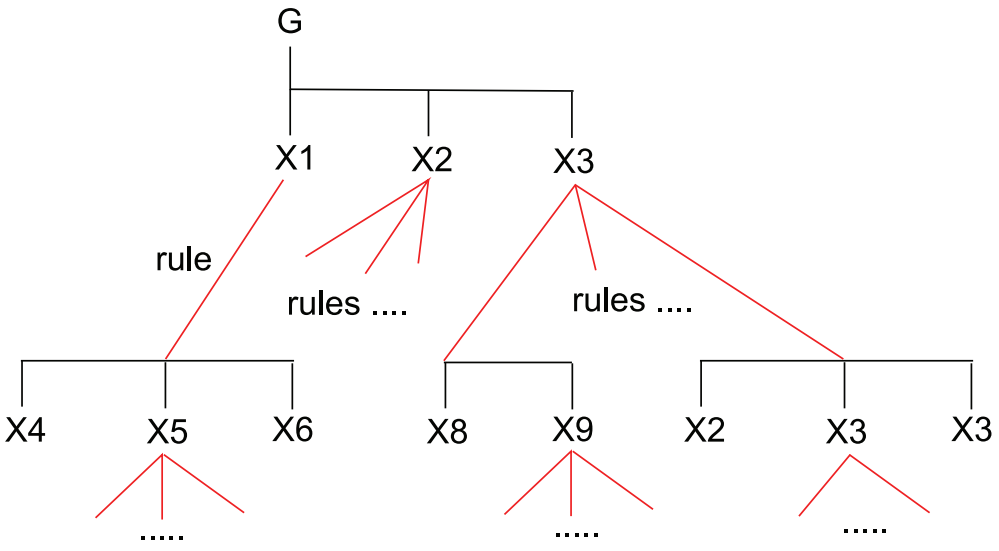
We are now able to work out the deduction algorithm. This algorithm can be very fast because:
1. the logic is truth-functional, ie, it ignores long-range probabilistic dependencies
2. the logic is algebraic, ie, it does not use the binary implication operator $\rightarrow$

Every rule in the logic is of the *conditional form*:
A $\leftarrow op$(B, C, D, ...)
where op is one of the operators described earlier.

The deduction algorithm will be given a query goal, G. It then seeks rules in the KB with G as the head:
G $\leftarrow op$(X1, X2, X3, ...)
and it basically performs recursion with X1, X2, X3, ... as subgoals.

Algorithm 2: simple backward chaining
Input: a knowledgebase KB, a query G
Output: the truth value of G
(1) **repeat forever**
(2) get a list of rules potentially applicable to G
(we maintain an index of predicates so we can quickly find the rules with the same head predicate as G)
(3) select an applicable rule R whose head unifies with G
(4) **if** recursion has gone too deep **or** the (incomplete) proof is already too long
(5) **return** 'fail'
(6) evaluate the rule R
(recurse to evaluate the variables in the rule body)

The search space of this algorithm is a so-called **AND-OR graph** (or tree) that often appears in logic-based AI search. Each AND is represented by a horizontal line (whose elements are the arguments of an operator and *all* of them must be evaluated); each OR is represented by a set of child branches (shown in red in the diagram) (whose elements are rules from the KB and only one needs to be selected). Thus the search is basically a matter of selecting which rules to apply.



Complexity. If m is the average number of rules applicable to a head predicate, n is the average number of arguments for each rule, and h is the depth of the search tree, then the size of the tree is $O((m \cdot n)^h)$. Let me make a wild guess that $m \approx 30, n \approx 5, h \approx 5$, then size $\approx 10^{11}$. This sounds manageable, but don't forget that the complexity of learning is the exponentiation of this! So it is important to keep things simple at this stage.

Efficient deduction.
1. Inference is often grounded by facts in **Working Memory** (ie, things that the AGI is currently paying attention to), though it may also be grounded by facts recalled from LTM (Long Term Memory). Perhaps an efficient search algorithm should *simultaneously* use backward chaining from the goal and forward chaining from ground facts under current attention.
2. Given a head predicate, the KB server can quickly retrieve a list of all rules with that predicate by looking up an index. Also, this list can be assumed to be sorted in order of confidence (remember that each rule is associated with a frequentist confidence). This order may provide a basis for best-first / heuristic search
2. Maybe the search should be randomized, and maybe it can start from the "middle" (ie, we generate an initial proof that is incorrect or incomplete, then repeatedly mutate it to make it correct)[4]. This suggests using an evolutionary algorithm, but EA's may be slow. { TO-DO: explore this idea further }

Solving this search problem is very important because inductive learning also depends on such a search.

5.3.5 Loopy inference

5.4 Abduction

(Logic-based) abduction and induction are closely related; they are 2 faces of the same coin. In both cases we seek a hypothesis H that together with background knowledge B, entails a positive example e^+:

$$B \cup H \vdash e^+$$

the only difference is that for abduction, H is ground (ie, contains no variables); and for induction, H is non-ground.

In classical logic, the algorithm for abduction is identical to deduction (backward chaining) except that the termination criterion is different. In deduction the leaves of the search tree should be ground facts in the KB (which are believed to be true). For abduction, the leaves should be so-called **abducibles**, which are propositions that can be assumed to be true. For example, "rain" is a common occurrence and thus can be assumed to be true in order to account for the grass being wet.

Under probabilistic logic, we no longer need the notion of abducibles. Interestingly, in this case the algorithm for abduction becomes almost identical to that of deduction, with the exception that deduction starts with an unknown fact whose truth is to be determined; whereas abduction starts with a known fact in need of explanations. The 2 search spaces are exactly the same.

TO-DO: unfinished.

[4] This idea is inspired by the GSAT and WalkSAT algorithms for propositional logic.

6 Pattern recognition

An example is the recognition of chairs by a Prolog rule such as:

```
chair :- 4-legs, seat, back.
```

In general the mechanism for pattern recognition is **forward-chaining** because we start with the premises (sensory input) and we do not know the desired conclusions in advance.

6.1 The theory-based theory

Concept formation (or "categorization" in the cognitive science literature, [Murphy, 2002], [Cohen and Lefebvre, 2005], [Margolis and Laurence, 1999], [Lakoff, 1987]) is the task of using machine learning to learn common-sense concepts ([Nakamura, Medin, and Taraban, 1993], [Wrobel, 1994]). [Wrobel, 1994] has summarized the following properties of human concepts:

1. concepts often have non-necessary features
2. disjunctive concepts (there may not be any features that are shared by all members of a concept)
3. relational information (seems to require first-order logic to represent)
4. some features are themselves concepts
5. typicality (people can often rank examples according to typicality, eg the most typical fruit is orange)
6. basic levels (people are more adapt at categorization at certain basic levels, eg naming an object "chair" rather than "office chair" or "a piece of furniture")
7. superordinate distance (eg "chicken" is rated more similar to "animal" than to "bird")
8. unclear cases (eg "is tomato a fruit?")
9. context-dependent effects
10. goal-dependent effects

There are two major theories of categorization: In the **Classical view** a concept is defined by a set of defining features which are individually necessary and sufficient. This view has very few adherents now. The other major theory is the **Exemplar view**, which classifies instances based on their similarity (eg a distance metric) to a set of existing exemplars.

In my opinion, also shared by [Murphy and Medin, 1985] and [Wrobel, 1994], the most satisfactory solution (aka the **theory-based theory**) is to view categorization as an *inference* process, where concept formation means constructing *explanations* of why certain objects belong to a concept.

6.2 Similarity

{ Similarity may be related to associative recall (§11.1), and may also be relevant to compression (§9.5) }

We're interested in measuring similarities between *structured* objects. This has been considered, eg in [Schmid, 2003], chapters 10-12, in an ILP setting. A famous example is the similarity between the solar system and the atom: the important point is that both systems have bodies revolving around a central body, but it does not matter what their sizes, masses, and temperatures, etc, are.

What we seek is a mapping between 2 structured objects, and to quantitatively measure the strength of such a mapping.

6.2.1 Motivating examples

(A) "Marmite is similar to Vegemite"

salty(marmite)	salty(vegemite)
dark-brown(marmite)	dark-brown(vegemite)
yeast-extract(marmite)	yeast-extract(vegemite)
rich-in(marmite, vitamin B)	rich-in(vegemite, vitamin B)
sub-string(name(marmite),"mite")	sub-string(name(vegemite),"mite")
popular-in(england)	popular-in(australia)

So I propose that the similarity between 2 objects can be calculated by considering all predicates that apply to both objects and obtaining the ratio between the number of identical and different predicates:

$$\text{similarity } \psi = \frac{N^{=}}{N^{=} + N^{\neq}} \tag{6.1}$$

{ TO-DO: If the predicates have complex structure, we need to (recursively) compare the other arguments, and if the latter are different, adjust for their differences. }

The above calculation can also be weighted by **information utility**, yielding the **subjective similarity** measure.

(B) "Lincoln is similar to Kennedy" [1]

elected-to-congress(lincoln, 1846)	elected-to-congress(kennedy, 1946)
elected-president(lincoln, 1860)	elected-president(kennedy, 1960)
succeeded-by(lincoln, vice-president$_1$)	succeeded-by(kennedy, vice-president$_2$)
name(vice-president$_1$, "johnson")	name(vice-president$_2$, "johnson")
born(vice-president$_1$, 1808)	born(vice-president$_2$, 1908)
assassinated$_1$(lincoln, friday)	assassinated$_2$(kennedy, friday)
in(assassinated$_1$, theater)	in(assassinated$_2$, car)
name(theater, "ford")	made(car, "ford")
with$_1$(wife(lincoln), lincoln)	with$_2$(wife(kennedy), kennedy)
during(with$_1$, assassination$_1$)	during(with$_2$, assassination$_2$)

Additionally:

name(car, "Lincoln")
has(kennedy, secretary), name(secretary, "Lincoln")

The additional facts do not increase the similarity, but they do make the 2 cases seem more "connected".

(C) Some quantitative examples:

the thing	approximated as	reason
orange T-shirt	red T-shirt	proximity in color space
7 people	several people	$\mathcal{Z}$
6'5 tall	very tall	$\mathcal{Z}$
(John lies on several occasions)	John often lies	$\mathcal{P}$
Stewart Shapiro	Stuart Shapiro	string edit distance

These examples can be represented and approximated by $\mathcal{P}/\mathcal{Z}$ values (eg with fuzzy pattern recognition).

(D) Some qualitative examples:

John is humorous $\approx$ John is witty
junk food, smoking and drinking $\approx$ unhealthy lifestyle
theft $\approx$ burglary
(complex scene) $\approx$ a fighting in a bar

[1] In this example I have used my own knowledge representation scheme Geniform. The example is based on a series of uncanny coincidences between the Lincoln and Kennedy assassinations that are often cited in trivia books.

7 Belief revision

Belief revision concerns the problem of conflicting conclusions and, in the extreme, contradictions.

One simple question is: If we arrive at two P(Z) distributions from two separate inference chains, how to combine the answers?

It seems that this is related to statistical inference. Suppose we have a population. A set of observations gives the estimate of the mean and variance:

$x_1, x_2, ..., x_{n_1} \sim Beta(\mu_1, v_1)$

Another set of observations gives another pair of estimates:

$y_1, y_2, ..., y_{n_2} \sim Beta(\mu_2, v_2)$

The question is to find a reasonable way to combine the 2 sets of observations to give a new estimate:

$x_1, x_2, ..., y_1, y_2, ... \sim Beta(\mu_0, v_0)$

In other words, how to express (μ_0, v_0) in terms of (μ_1, v_1) and (μ_2, v_2)?

Maybe the μ_0 is just the weighted average of μ_1 and μ_2? n_1, n_2 can be assumed to be equal to the supports w_1, w_2 which can be calculated from the confidences c_1, c_2.

Some assumptions in the above may not be entirely justified: the population size is typically small and the 2 samples may have overlap, ie, not independent.

7.1 Justifications

Justifications are the reasons why we believe in something.

Justification-based Truth Maintenance Systems (JTMS) keep track of justifications explicitly. They are needed because otherwise we would not be able to recall why certain beliefs are in the mind.

{ TO-DO: For example... }

This may be a requirement in addition to probabilistic reasoning.

7.2 Many-worlds representation

The test of a first-rate intelligence is the ability
to hold two opposed ideas in the mind at the same time,
and still retain the ability to function.
— F Scott Fitzgerald

With probabilistic logic, it may be possible to represent multiple possible worlds in a single KB, by storing multiple potentially-conflicting assumptions in the KB.

For example, in the *Truman Show* movie, there may be 2 competing assumptions in Truman's mind:

A1. Truman's world is normal.
A2. Turman's world is populated by fake actors.

In binary logic we can only accept either A1 or A2, thus the belief revision problem becomes unnecessarily much harder.

If we use probabilistic logic, then all we need is to impose that probabilities of alternative assumptions (ie, mutually exclusive or disjoint events) sum to 1. Disjoint means that $P(A \cap B) = 0$, ie, A and B cannot be true at the same time. The general rule is:

> If we have a set of assumptions such that $P(A_i \cap A_j) = 0$ for all $i \neq j$,
> then $\sum P(A_i) = 1$.

7.2.1 Deliberative assumptions and ATMS

Sometimes we make deliberative assumptions for more efficient reasoning, for example: *"Assuming the teacher will not check, I copy my classmate's homework"*.

This is in contrast to the kind of **implicit assumptions** described above, which are treated probabilistically, for example: "The teacher may check my homework with probability 0.2".

The difference is that we deliberately force the reasoner to speculate on the consequences of an assumption as if its probability is 1.

This kind of reasoning is similar to binary logic, in which we must be careful about keeping track of conflicting assumptions. **Assumption-based Truth Maintenance Systems** (ATMS) are developed specifically for this purpose.

ATMS seems somewhat redundant if we have probabilistic logic, where we can deal with multiple assumptions without probabilistic conflict (which in binary logic would result in conflicts).

If an assumption is particularly important, a probabilistic reasoner can take expected utilities into consideration.

7.3 Consistency check

A few types of consistency checks are needed.

7.4 Conflict resolution

Ie, what to do when an inconsistency is found.

7.5 Theory revision

See §9.2.

8 Meta-reasoning

I think I think, therefore I think I am.

The term "meta-reasoning" may refer to 2 things:
1. The ability to **reason about reasoning**, which is what this chapter is concerned with;
2. Scheduling reasoning tasks to achieve best results with limited computational resources (I have not thought about this problem yet).

An excellent survey of meta-reasoning in the #1 sense is [Constantini, 2002].

In the Tell-Learn loop, we see that we need a special predicate credible() to increase the probability of a statement via a side-effect. But that may not be the only meta-reasoning move we can make.

Another example is the $B \leftrightarrow Z$ conversion of "traitor" and "patriot" (§4.7).

8.1 Higher order logic

The HOL formulation that I am most familiar with is [Lloyd, 2003]. It uses **typed lambda calculus** to represent logical formulae. This representation is specifically developed for use as a logic programming language (Escher) and for inductive learning.

Interestingly, Lloyd uses a form of algebraic logic that is similar to what I do with $\mathcal{P}(\mathcal{Z})\mathcal{C}$ logic, ie, its statements are of the form $H = B$ where H is the head and B is the body. In essence this is the Prolog / Horn tradition.

{ TO-DO: formulate a $\mathcal{P}(\mathcal{Z})\mathcal{C}$ HOL }

9 Inductive learning

Rewards and punishment is the lowest form of education.
— Zhuangzi (4th Century BC)

Note: reinforcement learning is discussed in §12.4.

9.1 "Learn by being told"

This is probably the most efficient learning method. (As Martin Magnusson pointed to me, [Perlis and Purang, 1996] explored the possibility where a human is able to teach an AI via natural-language conversations. The paper contains hypothetical examples of some such conversations.)

The Tell-Learn loop. Translate a natural language sentence into a logical formula. Then the logical formula will directly enter the KB as knowledge, which is a very efficient way of learning.

The main difficulties in this loop are:

1. The assimilation of the fact into the KB, which requires belief revision
2. The translation from natural language to logic; sometimes this require abductive reasoning even when the NL parser does an adequate job
3. Surprisingly, inductive learning is also needed (see below)

The base logic should be:

1. simple, for machine learning
2. close to NL, for easy translation from NL to logic.

9.2 Background

Inductive learning using logic is studied under the heading ILP (inductive logic programming; an excellent and up-to-date survey of which is [Konstantopoulos, Camacho, Fonseca, and Costa, 2008]). The area known as theory revision is also essential to AGI (an excellent new book on ILP that discusses theory revision is [De Raedt, 2008]). { cite older books on ILP }.

Why is ILP needed? From my observations, when people use natural language to express ideas, they usually leave many **gaps** that must be filled by abductive or inductive reasoning. A parser can translate NL sentences into logic *literally*, but the resulting KB would be incapable of reasoning.

Filling in the logical gaps is very hard for humans because the inference of the gaps are inaccessible to conscious thinking. Therefore, an inductive machine learner is required to perform this function.

9.3 Examples

A. "Women usually have long hair"

Example facts:

Mary is a girl, Mary has long hair
Jane is a girl, Jane has long hair
John is a guy, John has short hair
etc...

To learn:

female(X) $\rightarrow$ long-hair(X)

This example illustrates the **MDL principle**: The general rule *covers* many examples and thus is *compressive* – In some situations, we may forget individually which woman has long hair, while remembering a group of women as mostly having long hair (eg, after seeing a group of choir singers).

We can make this example more complex by adding the role of a speaker:

The teacher tells me "Mary has long hair"
The teacher tells me "Mary is a girl"
The teacher tells me "Jane has long hair"
The teacher tells me "Jane is a girl"
etc...

We can have the general rule: [1]

"If a speaker is a credible source, then what s/he says can be assumed to be true"

R1: credible(X) $\wedge$ says(X,Y) $\rightarrow$ Y

But notice that R1 can cover the girls examples only if we include the quoted sentences:

R1 $\wedge$ Y=long-hair(mary) $\vdash$ long-hair(mary)
R1 $\wedge$ Y=female(mary) $\vdash$ female(mary)
R1 $\wedge$ Y=long-hair(jane) $\vdash$ long-hair(jane)
R1 $\wedge$ Y=female(jane) $\vdash$ female(jane)

Thus, we can further compress the right hand sides with this rule:

R2: female(X) $\rightarrow$ long-hair(X)

or compress what the teacher says:

says(teacher,"female(X) $\rightarrow$ long-hair(X)")

This shows that even when an example is explained (entailed) by a general rule (R1), it may still require compression (R2). This is different from the traditional ILP goal which is simply to "cover" (ie entail) positive examples. Now we need to find *all possible* ways to compress the KB.

B. Roman numerals

Example facts:

Roman numeral 1 is I
Roman numeral 2 is II
Roman numeral 3 is III

To learn:

Roman numeral n is n I

(which is wrong, but is reasonable in common sense).

9.4 Complexity

[1] In this example we have used 3 tricks:
a. propositions as variables
b. quoted formulae
c. equality
just for the sake of illustration. The actual KR scheme is still undecided.

9.4.1 Induction of one hypothesis

The main task of ILP is to search in the hypothesis space $\mathbb{H}$ specified partly by the logical syntax (such as P(Z) logic) and partly by background knowledge (eg what kind of predicates are present). The hypothesis space is structured into a lattice with a partial order (usually denoted $\preceq$). The partial order can be one of θ-**subsumption**, LGG (**least general generalization**), or **logical entailment**. These have subtle differences, with logical entailment being the most rigorous. The searcher can traverse up and down the lattice, corresponding to **generalization** and **specialization** of the hypothesis. These 2 are known as **refinement operators**. Because there are so many ways to generate hypotheses in FOL (with many predicates to choose from, and the possibility of introducing variables as desired), the branching factor is extraordinarily high. Also, the lattice can be infinite, so we need to limit its depth.

Another problem is that, once a hypothesis has been generated, it must be tested against the rest of the KB for **consistency**. The consistency check must also be limited in depth, but even then it is very time-consuming.

Much of the complexity of induction occurs in the "inventive step", whereby a new rule is generated. The inventive step can draw upon knowledge from the entire KB, which makes it combinatorially explosive. Currently the most promising idea I have is to "recall similar examples" – that means, we perform an **associative memory search** to recall examples similar to the current experience, and only then do we try to generalize from the experience plus the recalled examples. But this strategy merely pushed the responsibility to the associative memory to recall only the most **salient / relevant / interesting** examples. Still, this trick seems to be more efficient than blind search.

Another promising idea is that some of the more intensive processing can be performed during **sleep**.

9.4.2 Induction of a whole theory

Let $\mathbb{H}$ denote the hypothesis space (ie, space of all logic formulae). The size of $\mathbb{H}$ is huge (see above).
A logical theory T is a set of hypotheses, $\{h_i | h_i \in \mathbb{H}\}$. Note that $T \in 2^{\mathbb{H}}$ which is a hugely huge space.
Let E denote the set of examples $\{e_1, ...\}$ that should be covered by the target theory.
So we're seeking an optimal theory T^* such that it covers all examples and is succinct, ie:

$$T^* \vdash \{e_i\} \quad \text{and} \quad |T^*| \text{ is minimal.}$$

There is hope, since we can *seed* the initial theory with human knowledge. In other words, we can start with a T_0 which contains a large number of logic statements translated from NL.

Even if we do that, we can realistically only store one theory in memory. When we evaluate a new hypothesis h to cover an example e, ie, to test whether

$$h \cup T \vdash e \, ?$$

we have to bear in mind that we are just testing the hypothesis h w.r.t. *one* background theory. This means that h may have a different score when T changes; but unfortunately we cannot realistically keep track of different scores of h against different theories. In other words, the scores we keep will be somewhat *inaccurate* and are *relative* to a constantly changing candidate theory. { TO-DO: Due to the use of probabilistic logic, we can store multiple sets of hypotheses simultaneously. Revise this paragraph. }

9.5 Compression

Usually, an example can be covered (entailed) by a number of alternative proofs. This is probably very common. For example:

Mary doesn't love John $\vdash$ John is unhappy
John doesn't love Kate $\vdash$ Kate is unhappy
Pete has no money $\vdash$ Pete is unhappy

but we can further compress the right hand sides by:

everyone is unhappy

even though the right hand sides were entailed by 3 different reasons.

Having a large number of general rules enables better **reconstruction** of past memories. Better reconstruction is clearly an advantage from a utilitarian perspective as we often do not know in advance what memories may be useful later. (Overlapping coverage may be a by-product of having a large number of rules.)

We can define **information utility** as the utility of information items ($U : \{infon\} \to \mathbb{R}$) [2] , as a generalization of utility over states ($U : \{state\} \to \mathbb{R}$). Then the goal of compression is to *minimize reconstruction errors (weighted by information utility) with the smallest theory*. This is in keeping with the MDL principle.

Formally, the goal is to find a theory T to compress a set of examples $E = \{e_i\}$ such that:

$$\begin{cases} T \vdash \hat{E} & \text{(reconstruction of memory)} \\ \hat{E} \approx E & \text{(approximation)} \\ |T| \leq |E| & \text{(smaller description length)} \end{cases} \tag{9.1}$$

Note: the entailment $\vdash$ should be fuzzy/probabilistic.

In general, a logic based system performs compression via **pattern recognition** (§6). Upon the arrival of new sensory input, the Pattern Recognizer is invoked to recognize patterns in the data (essentially the same as forward-chaining). Due to the use of **fuzzy** pattern recognition, the result may be lossy, which is good.

The job of the Inductive Learner is to generalize from regularities in the sensory input whenever possible, adding new rules to the KB which will be used by the Pattern Recognizer to recognize new patterns in the future.

In traditional ILP with binary logic, we settle at finding a theory that covers all positive examples and none of the negative examples. In the compression setting, mere coverage is not enough: we should generate new rules even when the data are already entailed by some existing rules, because the new rules can further compress data.

9.5.1 Plateau phenomenon (local minima)

Some changes in T may increase the error first but further changes in that direction may result in a smaller error. This phenomenon (ie, local minima) is common in ILP and makes learning more difficult.

An example is given in [Bergadano and Gunetti, 1996], p89, for the learning of the Prolog function `append/3`. The positive and negative examples are:

e^+: `append([0],[1,2],[0,1,2])`
`append([],[1,2],[1,2])`
e^-: `append([0],[1,2],[0,1])`
`append([0],[1,2],[1,2])`
`append(0,1,[0,1])`
`append(a,b,[a,b])`

And the target clause is:

```
append(X,Y,Z) :- list(X), head(X,X1), tail(X,X2), append(X2,Y,W), cons(X1,W,Z)
```

The learning of the first literal, `list(X)`, would indeed have positive information gain [3] as it excludes the last 2 negative examples; but the addition of the following 3 literals would have small or even negative information gain, until the last literal is reached and all positive and negative examples are covered.

9.6 Minimum description length

An interesting observation (noted in [MacKay, 2003] Chapter 28, and [Chater, 2005]) is the equivalence of MDL and Bayesian likelihood maximization: Given the data D, The goal of perception is to choose the hypothesis H^* that maximizes $P(H|D)$. According to Bayes theorem, this is equivalent to choosing the H^* that maximizes $P(D|H)P(H)$, where $P(H)$ represents some sort of subjective prior bias. The H^* that maximizes $P(D|H)$ is the same H^* that minimizes the negative of the logarithm of this term, which can be written:

$$-\log_2 P(H) - \log_2 P(D|H).$$

By Shannon's coding theorem, the optimal code length for a probability p is $-\log_2 p$, so we can interpret the above as:

$$\text{codelength}(H) + \text{codelength}(D|H).$$

Note: the MDL terminology is different from the logic terminology:

[2] An infon is informally defined as a discrete information item. In practice it can be any proposition in the KB.
[3] The definition of information gain is that used by the ILP program FOIL. See the book for details.

in MDL	in logic
a hypothesis	is called a logical theory
N/A	a hypothesis (= part of a logical theory)

{ TO-DO: How does this shed light on the logical compression algorithm? }

9.7 Algorithm

(Cf algo 1 for the top-level algorithm of the logical reasoner.)

The goal of the learning algorithm is to produce useful generalizations, or we can say the most *compressive* generalizations.

Algorithm 3: Practical inductive learner
Input: incoming facts
Output: 1 or more hypotheses
(1) Try to recall similar examples from memory.
(2) Invent a hypothesis to generalize the new fact + examples

I look at the problem from a perspective that is slightly different from the one prevalent in the ILP literature. Most ILP methods search in the *hypothesis space*, but I find it easier to think about the problem in *proof space*:

- My algorithm searches for explanations of *one incoming fact* while inventing (possibly many) hypotheses along the way.
- The hypothesis-space algorithm, on the other hand, focuses on *one hypothesis*. It would pick some examples (past or incoming facts) and move around the hypothesis lattice in order to cover them, ie, keeping a score of the numbers of positive and negative examples that are covered or not. And it seeks the hypothesis with the best score.

10 Natural language

The fish trap exists because of the fish; once you've gotten the fish, you can forget the trap. The rabbit snare exists because of the rabbit; once you've gotten the rabbit, you can forget the snare. Language exists because of meaning; once you've gotten the meaning, you can forget language.
— Zhuangzi (4th Century BC)

10.1 Natural language is not essential to AGI

Needless to say, fully solving the natural language problem is AGI-complete. This however is not a show-stopper. Our ultimate goal is to reach the **RSI point** (§1.4) with as little effort as possible. Which means we only need the most basic system that can automatically write programs for us. Such a system only needs to accept commands in a *restricted* subset of English. Thus, for all our purposes an NL module that can parse restricted English would suffice.

10.2 Unification-based grammars

The family of unification-based grammars includes LFG (Lexical Functional Grammar), HPSG (Head-Driven Phrase Structure Grammar), and PATR grammar. The unification algorithm used in unification-based grammar is the same as the unification algorithm used in logic. This is further evidence that the brain employs first-order symbolic processing.

10.3 Cognitive linguistics

Currently we are using the "formal semantics" approach.

{ TO-DO: How may cognitive linguistics affect the design of the natural language subsystem? }

10.4 Abduction as interpretation

10.4.1 A detailed example

The general sequence is:
tokenization → POS-tagging → syntax parsing → semantic parsing
which should be familiar to everyone with experience in NL processing.

I will explain the details using a simple example:
"I love Mary"

The crucial thing is that we represent everything in a logic-based framework. First we represent the sentence as "raw data" (ignoring tenses to simplify matters):

sentence(e_0)
lexeme-I(e_1)
lexeme-love(e_2)
lexeme-Mary(e_3)
begins-with(e_0, e_1)
follows(e_2, e_1)
follows(e_3, e_2)

where:

the e_i's are **entities** (logical constants).
the entities e_1, e_2, and e_3 are words.
follows() means a word follows another word in a sentence.

Up to now, all we have is a sentence as raw text (without meanings). The next step is to recognize parts of speech (nouns, verbs, adjectives, etc). We can use logical rules to do this.

An example logical rule is:

lexeme-Mary(X) $\rightarrow \exists e$ parse-as(e, X) $\wedge$ noun(e)

which simply means that the word "Mary" is a noun. X is a variable (implicitly universally quantified). e is a new entity, which instantiates to e_4 when the rule is applied.

We can also use logical rules to parse syntax. We can perform "VP $\leftarrow verb + noun$" with this rule[1]:

verb(X_1) $\wedge$ noun(X_2) $\wedge$ follows2(X_2, X_1) $\rightarrow \exists e$ parse-as(e, X_1, X_2) $\wedge$ $vp(e)$

which creates a new entity e_5 which is a VP.

Assume that eventually we have a parse of the sentence S, e_6. Notice that up to now, it's all syntax parsing.

Next we perform semantic parsing. The key is to generate partial meanings for phrases, such as the verb phrase "loves Mary".

Lambda operator. In formal semantics, it is customary to use the λ operator to represent the meaning of phrases. The reason is that first-order logic do not have the expressive power to represent such phrases. For example, the VP "loves Mary" denotes "somebody's loving Mary", which may be represented as $love(\_, mary)$; but that is not a well-formed formula in FOL. Instead we can represent it using the λ expression $\lambda x\ love(x, mary)$. This is no longer FOL, but is a so-called *quasi-logical* form. I propose to use an alternative method which is within FOL. It employs the composition functor.

Composition functor. (See also §3.3) Under this method, all phrases are represented by compositions. For example, the VP "loves Mary" can be represented by $comp(loves, mary)$, or in short-hand $loves * mary$. This composite is a *first-order object*, which cannot be further reduced, but when we compose it with $john$, we get $john * (loves * mary)$ which is equivalent to the first-order statement $loves(john, mary)$. So we need special inference rules to convert such composites into statements.

I think this method is superior to λ because the meanings of phrases can be represented by first-order objects. It also makes semantic parsing very intuitive.

{ TO-DO: Explain semantic parsing with examples. }

10.5 Universal logical form

An immediate question is how to translate "nearly every text sentence under the sun" into logical form. To which we employ several rules to quickly reduce the problem space:

1. Every noun (except special nouns such as pronouns) corresponds to a concept which is represented by a predicate. For example, the word:chair corresponds to the concept:chair and is represented by the predicate `concept-chair()`.

[1] We need this special rule:
follows(X_1, X_2) $\wedge$ parse-as(X_1, X_3) $\wedge$ parse-as(X_2, X_4) $\rightarrow$ follows2(X_3, X_4)

10.6 Some example English sentences

11 Memory systems

Computer science has only three ideas: cache, hash, trash
— Greg Ganger, CMU

11.1 Associative memory

Associative recall can greatly facilitate inductive learning.

Sometimes fuzzy associations may be desirable.

The KB being in first-order logic poses problems for fuzzy associations. It is unlike associative neural networks which may be closer to propositional logic.

11.2 Efficient rule selection

{ TO-DO: This idea is problematic. It only works if the approximate oracle is correct a high percentage of the times; but this is highly suspect. }

Given:
1. The goal (ie, the conclusion of the proof)
2. The premises (ie, facts residing in Working Memory)
Try to predict:
3. The rules involved in the proof

Each data point would be:
1. goal (a grounded fact)
2. premises (set of grounded facts)
3. a critical rule

So it seems:
$\{ \text{ fact } \} \times \{ \text{ set of facts } \} \rightarrow \text{rule}$

Using multidimensional scaling, we can map a fact to its coordinates in a high-dimensional space.

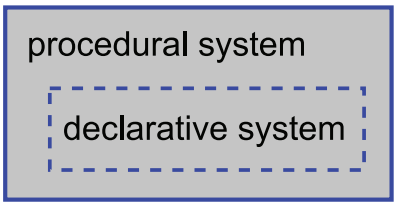
Then we can partition the high-dimensional space into grids, and to each grid assign a bucket of rules. We need some way to partition the high-dimensional space. But we can also partition the space of data points into many categories?

12 Planning and acting

12.1 "Procedural subsumes Declarative"

So far, we have only talked about the **declarative** aspect of AGI. Now we need to consider the **procedural** aspect.

It seems natural that the procedural system should subsume the declarative system:



The procedural system communicates with the declarative system via various types of **querying**. The declarative system itself has no means of performing actions; its only function is question-answering. There may be exceptions, but the general rule can be captured by the slogan *"Procedural subsumes Declarative"*.

12.2 The action language

We have represented declarative knowledge using a logical language; now we should represent procedural knowledge with a similar action language.

The following are examples of "actions":
1. say "hello"
2. open a file
3. print 1...100
4. repeat printing N until the user presses the Esc key

Do they sound like programming language tasks? Thinking along this line, it seems advantageous that *the action language should be a conventional programming language* such as C++, Java, Lisp, Prolog, etc.

So the procedural system expresses actions in a programming language. Those actions can be executed via the compiler or interpreter of that language. In some situations, it is more convenient if the language can be interpreted.

The action output of the Procedural System can be of 3 forms:
1. a statement that is immediately interpreted and executed
2. a program that needs to be compiled
3. a program fragment that becomes part of the Procedural System

#3 is actually a form of procedural learning.

12.3 Procedural learning

When the procedural language is complex, learning may be more difficult. So there may be a need to restrict the procedural language.

Learned procedures can be inserted into the Procedural System at certain hook points. Each hook point represents a context, for example "We get here when the user presses the Esc key". The procedural learning algorithm should take such contexts into consideration.

"learn by being told" is also applicable to procedural learning.

12.4 Reinforcement learning

The goal of RL is to learn an optimal policy which is a function from the set of states to the set of actions:
$\pi : \mathbb{S} \rightarrow \mathbb{A}$.

RL can learn to:
1. conversate with humans (eg in chat rooms)
2. write programs
3. crawl the web to learn things

and do all these without the need for human programming, so it is cost-effective.

See also §12.6.2 on combining deductive planning and RL.

12.4.1 Relational reinforcement learning

(cf [Dzeroski, de Raedt, and Blockeel, 1998], [Dzeroski, de Raedt, and Driessens, 2001], [Tadepalli, Givan, and Driessens, 2004], [Driessens, 2005])

One problem with RL is that the number of states in a general intelligent agent is too large (in fact, infinite). I suggest using FOL to describe the states so the formulation can be more compact.
A state would be a conjunction of ground literals, such as:

$$S1 = \text{horny(john)} \wedge \neg\ \text{good-looking(john)} \wedge \text{has-money(john)}$$

and the policy could be a rule that recommends an action from a state:

$$\neg\ \text{good-looking(X)} \wedge \text{has-money(X)} \rightarrow \text{try(X, buy-a-cybernetic-body)}$$

Such a logical rule is not exactly a function, because:

1. More than one state can be true at the same time
2. If 2 states are both true and S2 $\supset$ S1, and
   $S1 \rightarrow$ try(action1)
   $S2 \rightarrow$ try(action2)
   then both actions will be recommended when S2 is true. We need a **conflict resolution** scheme to resolve this.
   { TO-DO }

12.5 Means-end analysis

MEA is a planning method proposed first by [Newell, Shaw, and Simon, 1960] (General Problem Solver), who also believed that MEA occurs in human problem solving. Its general idea is similar to forward chaining state-space search (applying operators forwardly until the goal state is reached). MEA selects a difference between the current state and the goal state, selects an operator may reduce the difference, and attempts to apply the operator. If the operator's preconditions are not met, MEA recursively calls itself to transform the current state into one that meets those conditions. If the conditions are met, MEA applies the operator and then recurse to transform the new state into the goal state (from [Langley and Allen, 1993]).

Wikipedia: "Note that, in order for MEA to be effective, the goal-seeking system must have a means of

associating to any kind of detectable difference those actions that are relevant to reducing that difference."

{ TO-DO: how to combine MEA with RL and deductive planning? }

12.6 Deductive planning

A classical planning problem is represented by states and operators (actions). Each operator has its precondition and effect.

In deductive planning based on situation calculus, actions are represented as terms, with the special predicates `do()` and `poss()`.

An advantage of deductive planning is that the Procedural System only needs to be a inference engine and nothing else.

One difficulty here is how to represent states, actions, preconditions, and effects; especially actions. I can put the cup on the table. The current state is represented implicitly by all the facts in KB. The actions are a special class of facts related to possibilities. "Water conducts electricity" is a fact; But "water can conduct electricity" is a possible action?

Also, DP has a problem in that it only pursues one goal at a time.

12.6.1 Example

> *I try to kill Frank by tossing a radio in the bathtub while he is in it. This may work because I know that water conducts electricity, and electricity can kill, etc.*

Deductive planning allows the AGI to draw general knowledge from the KB to represent the planning problem:

goal = kill Frank
current state = Frank in bathtub, radio nearby
possible actions = toss radio into bathtub, slash Frank's wrist with a knife, etc
background knowledge = water conducts electricity, etc

12.6.2 Combining reinforcement learning and deductive planning

DP says: "If I do X, Y will probably happen as a result".
RL says: "At state S, if I perform action A, the reward is probably high".

We cannot simply have 2 planners co-existing – the actions recommended by DP may conflict with those recommended by RL. We may simply stipulate that the logical reasoner (since it is cognitively more sophisticated) has priority over RL.

RL may invoke DP for a recommendation of action. When DP fails to find a recommendation, RL will act.

13 Program synthesis

The most critical milestone of our project is to create a system that can perform simple automated programming. Traditional program synthesis systems are hard to use because:
1. They require *formal specifications* of program requirements, which are often as hard to write as the programs themselves, sometimes even harder.
2. The proof search is *too slow* for any problem of practical size, or the systems often require human interaction for guidance.

My proposal to solve these two problems are, respectively:
1. Allow **informal** specification of programming goals. This means using (restricted) natural language, including vague / probabilistic language.
2. In order to speed up the program search, it seems that the only viable solution is to use knowledge to guide the search. This is something that has been recognized in the AI community for a long time — no clever search algorithm or heuristic can improve the search significantly, without domain-specific background knowledge. Machine learning can be used to acquire such knowledge, but it is often an extremely difficult problem in itself; and we are also talking about a large body of background knowledge. As I have argued in §9, the most efficient way to acquire knowledge is to give up machine learning and simply **learn by being told**.

As with many AI problems, the program synthesis problem can be construed as a search, and we have the usual choice of top-down and bottom-up.

13.1 Formal program synthesis

14 Value judgments

The real question is not whether machines think but whether men do.
— B F Skinner

14.1 My stance on AGI friendliness

I do not have a rigorous theory for AGI friendliness and I'd be very suspicious of any such theory. But I believe that AGI technology should be made widely available to the general public, because the distribution of power is the best safeguard against abuses of AGI.

Other than that, the AGI should be subject to certain laws that prevent exploitation (eg, unlimited growth or taking up of physical resources). That would be the job of governments.

14.2 Sentient vs non-sentient AGI

15 Vision and other senses

TO-DO: transfer web pages to here.

15.1 Logic-based vision

16 Implementation

The sooner you start coding your program the longer it's going to take
— H F Ledgard 1975

Premature optimization is the root of all evil.
— Donald Knuth

The First Rule of Program Optimization: Don't do it.
The Second Rule of Program Optimization (for experts only!): Don't do it yet.
— Michael A Jackson

16.1 Choice of programming language

Lisp is good for rapid prototyping.

Prolog seems not as good as Lisp because it imposes the use of SLD resolution and depth-first search strategy.

Haskell may be slow because of lazy evaluation.

Low-level languages:

Object-orientation is not particularly natural for some software architectures.

Java is preferable to C# for its cross-platform maturity.

C may be too old-fashion.

C++? Not bad in my opinion.

Acknowledgments

In addition to the people listed on the title page, I'd like to thank the AGI mailing-list participants for years of discussions.

Glossary

ATMS assumption-based truth maintenance system

ATP automated theorem proving

BDI belief-desire-intention (architecture)

CDF cumulative distribution function

CNF conjunctive normal form

DNF disjunctive normal form

DP deductive planning

EA evolutionary algorithm

FOL first-order logic

ground In logic, a formula is ground if it does not contain variables

HO higher-order

HOL higher-order logic

ILP inductive logic programming

JTMS justification-based truth maintenance system

KB knowledge base

KR knowledge representation

LBAI logic-based AI

LGG least general generalization

literal In logic, a literal is an atomic formula or its negation

LTM Long-Term Memory

MDL minimum description length

MEA means-end analysis

NL natural language

NP noun phrase

NR natural reasoning (= common-sense / human-like reasoning)

PCA principal component analysis

PDF probability density function

RL reinforcement learning

RSI recursive self-improvement

rule a logic formula with variables (opposite of ground)

SAT logical satisfiability

SLD selective linear definite clause resolution

SLS stochastic local search

SVM support vector machines

term In logic, terms are the arguments of predicates; a term can be a variable, a constant, or recursive applications of functions to terms

TV truth value

VP verb phrase

WM Working Memory, part of the KB that has faster processing speed

ZOL zeroth-order logic (= propositional logic)

Bibliography

Akman, Surav, and Giunchiglia. Steps toward formalizing context. *AI magazine*, 17:55–72, 1996.

Baum. *What is thought?* MIT press, 2004.

Bergadano and Gunetti. *Inductive logic programming - from machine learning to software engineering*. MIT, 1996.

Bergmann. *An Introduction to Many-Valued and Fuzzy Logic*. Cambridge university press, 2008.

Bier. When uncertainty is irrelevant to expected utility decision making. *Uncertainty Modeling and Analysis, 1993. Proceedings., Second International Symposium on*, pages 401–407, 1993.

Boole. *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Walton and Maberley, London (reprint Dover, New York, 1958), 1854.

Bosc and Prade. An introduction to the fuzzy set and possibility theory-based treatment of flexible queries and uncertain or imprecise databases. In Motro & Smets, editor, *Uncertainty in Information Systems: From needs to solutions*, pages 285–324. Kluwer, Boston, MA., 1997.

Bratman. *Intentions, plans, and practical reason*. 1987.

Bratman, Israel, and Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4: 349–355, 1988.

Campbell. The sorites paradox. *Philosophical Studies*, 26:175–91, 1974.

Chandru and Hooker. *Optimization methods for logical inference*. Wiley interscience, 1999.

Chater. *Advances in minimum description length*, chapter 15 "A minimal description length principle for perception", page 385ff. MIT, 2005.

Cohen and Lefebvre, editors. *Handbook of categorization in cognitive science*. Elsevier, 2005.

Constantini. Meta-reasoning: a survey. *Computational logic: logic programming and beyond, LNCS*, 2408:65, 2002.

L. De Raedt. Logical and relational learning. 2008.

Domingos and Richardson. Markov logic: a unifying framework for statistical relational learning. In Getoor and Taskar, editors, *Statistical relational learning*. MIT, 2007.

Domingos, Poon, and Sumner. A general method for reducing the complexity of relational inference and its application to mcmc. *Proceedings of the 23th AAAI conference on AI*, pages 1075–1080, 2008.

K. Driessens. Relational reinforcement learning. *AI Communications*, 18(1):71–73, 2005.

Dzeroski, de Raedt, and Blockeel. Relational reinforcement learning. *Proceedings of the 15th international conference of machine learning*, pages 136–143, 1998.

Dzeroski, de Raedt, and Driessens. Relational reinforcement learning. *Machine learning*, 43:7–52, 2001.

Edgington. Validity, uncertainty and vagueness. *Analysis*, 52:193–204, 1992.

Fine. Vagueness, truth and logic. *Synthese*, 30:265–300, 1975.

Garcez, Lamb, and Gabbay. *Neural-symbolic cognitive reasoning*. Springer, 2009.

Gartner. *Kernels for structured data*. World scientific, 2008.

Getoor and Taskar, editors. *Statistical relational learning*. MIT, 2007.

Getoor, Friedman, Koller, Pfeffer, and Taskar. Probabilistic relational models. In *Statistical relational learning*. MIT, 2007.

Goertzel and Pennachin. *Artificial general intelligence*. Springer, 2007.

Goertzel, Ikle, Goertzel, and Heljakka. *Probabilistic logic networks – A Comprehensive Framework for Uncertain Inference*. Springer, 2008.

Graff and Williamson, editors. *Vagueness*. Ashgate, Dartmouth, 2002.

Haddawy. Generating bayesian networks from probability logic knowledge bases. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*. 1994.

Hailperin. Best possible inequalities for the probability of a logical function of events. In *American Mathematical Monthly*, volume 72, pages 343–359. 1965.

Hansen, Jaumard, de Aragao, Chauny, and Perron. Probabilistic satisfiability with imprecise probabilities. *International Journal of Approximate Reasoning*, 24:171–189, 2000.

Jaeger. Relational bayesian networks. *Proceedings of the conference on uncertainty in artificial intelligence*, 1997.

Jaumard and Parreira. An anytime deduction algorithm for the probabilistic logic and entailment problems. preprint submitted to Elsevier, 2006.

Johnson-Laird. *Mental Models*. Harvard University Press, 1983.

Keefe. *Theories of Vagueness*. Cambridge university press, 2000.

Kenevan and Neapolitan. *Fuzzy logic for the management of uncertainty*, chapter A model theoretic approach to propositional fuzzy logic using Beth Tableaux, pages 141 – 157. John Wiley & Sons, Inc. New York, NY, USA, 1992.

Kersting and de Raedt. Bayesian logic programs. In Cussens & Frisch, editor, *Work-in-progress reports of the 10th international conference on inductive logic programming (ILP 2000)*. 2000. URL http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-35/.

S. Konstantopoulos, R. Camacho, N.A. Fonseca, and V.S. Costa. Induction as a Search Procedure. *Artificial Intelligence for Advanced Problem Solving Techniques*, page 166, 2008.

R. Kurzweil. *The singularity is near: When humans transcend biology*. Viking, 2005.

Lakoff. *Women, Fire, and Dangerous Things - what categories reveal about the mind*. University of Chicago Press, 1987.

Langley and Allen. *Machine learning methods for planning*, chapter 10, A unified framework for planning and learning, page 317ff. Morgan Kaufmann, 1993.

Laskey. Mebn: a logic for open-world probabilistic reasoning. *GMU C4I Center Technical Report C4I-06-01*, 2006.

Lloyd. *Logic for learning*. Springer, 2003.

MacKay. *Information theory, inference, and learning algorithms*. Cambridge, 2003.

Margolis and Laurence, editors. *Concepts - core readings*. MIT press, 1999.

Milch, Marthi, Russell, Sontag, Ong, and Kolobov. Blog: probabilistic models with unknown objects. In *Statistical relational learning*. MIT, 2007.

Muggleton. Stochastic logic programs. In de Raedt, editor, *Advances in inductive logic programming*, pages 254–264. IOS Press, Amsterdam, 1996. URL http://www.doc.ic.ac.uk/~shm/Papers/slp.pdf.

Muggleton, Lodhi, Amini, and Sternberg. Support vector inductive logic programming. *Discovery science, LNCS*, 3735:163–175, 2005.

Muresan. *Scale Independence in the Visual System*, pages 1–18. Springer, 2004. URL http://www.raulmuresan.ro/Papers/MuresanSpringer.pdf.

Murphy. *The Big Book of Concepts*. MIT press, 2002.

Murphy and Medin. The role of theories in conceptual coherence. *Psychological review*, 92(3):289–316, 1985.

Nakamura, Medin, and Taraban, editors. *Categorization by humans and machines*. The psychology of learning and motivation, no 29. Academic Press, 1993.

Neapolitan. *Learning Bayesian networks*. Prentice Hall, 2004.

Newell, Shaw, and Simon. Report on a general problem solving program for a computer. *Proceedings of the international conference on information processing*, pages 256–264, 1960.

Ng and Subrahmanian. Probabilistic logic programming. *Information and computation*, 101(2):150–201, 1992.

Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

Parsons. *Qualitative methods for reasoning under uncertainty*. MIT, 2001.

Pearl. *Probabilistic Reasoning in Intelligent Systems Networks of Plausible Inference*. Morgan Kaufmann, California, 1988.

Perlis and Purang. Conversational adequacy: Mistakes are the essence. *Int. J. Human-Computer Studies*, 48: 553–575, 1996.

Russell and Norvig. *Artificial Intelligence - a modern approach*. Prentice Hall, 2003.

Sainsbury. Degrees of belief and degrees of truth. *Philosophical Papers*, 15:97–106, 1986.

Sato and Kameya. Prism: A language for symbolic-statistical modeling. *IJCAI*, pages 1330–1339, 1997. URL http://sato-www.cs.titech.ac.jp/prism/.

Schmid. *Inductive synthesis of functional programs*. Springer, 2003.

Shapiro. *Vagueness in Context*. Oxford university press, 2006.

Sowa. *Knowledge representation - logical, philosophical, and computational foundations*. Brooks/Cole, 2000.

Stickel. A prolog technology theorem prover: implementation by an extended prolog compiler. *Journal of Automated Reasoning*, 4:4:353–380, 1988. URL http://www.ai.sri.com/~stickel/pttp.html.

Sundgren, Danielson, and Ekenberg. Some second order effects on interval based probabilities. pages 848–853, 2006.

P. Tadepalli, R. Givan, and K. Driessens. Relational reinforcement learning: An overview. In *Proceedings of the ICML*, volume 4, pages 1–9, 2004.

Thornton. Parity: the problem that won't go away. *McCalla (Ed.), Proceeding of AI-96 (Toronto, Canada)*, pages 362–374, 1996. URL http://www.cogs.susx.ac.uk/users/christ/papers/parity-here-to-stay.pdf.

Thornton. *Truth from trash — how learning makes sense*. MIT, 2000.

Walley. *Statistical reasoning with imprecise probabilities*. Chapman and Hall, 1991.

Wang. Reference classes and multiple inheritances. *Technical Report No. 91, The Center for Research on Concepts and Cognition*, 1994. URL http://www.cogsci.indiana.edu/farg/peiwang/papers.html#reference_classes.

Wang. *Rigid Flexibility The Logic of Intelligence*. Springer applied logic series, 2006.

Wellman, Breese, and Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7 (1):35–53, 1992.

Wikipedia, 2008. URL http://en.wikipedia.org/w/index.php?title=Integration_by_substitution&oldid=220988927.

Williamson. *Vagueness*. London: Routledge, 1994.

Wrobel. *Concept formation and knowledge revision*. Kluwer academic press, 1994.

Zabell. We johnson's 'sufficientness' postulate. *The annals of statistics*, 10:4, 1982.

Zadeh. Fuzzy sets. *Information and control*, 8:338–353, 1965.