Genifer
– an artificial general intelligence

YKY (general.intelligence@Gmail.com)

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

0 Preface, executive summary, to-do list

1. This book is a perpetual draft.
2. My personal reason for developing AGI is to achieve life extension.
3. The source code of Genifer is hosted on Google Code, including some very easy tutorial slides. Also feel free to contact me!

— YKY

Executive summary:

Inference: Genifer descended from classical logic-based A.I. Its 3 modes of inference are deduction, abduction (explaining), and induction (learning). This is common to NARS, OpenCog, Cyc.

Logic: Genifer is based on an **algebra of concept composition**, which replaces predicate logic as the internal structure of propositions.

KB: Genifer's KB stores logic formulas, similar to classical A.I. systems such as Cyc, and NARS. OpenCog is an exception in that it stores its knowledge as a hypergraph called AtomSpace.

Uncertainty: Genifer uses fuzzy-probabilistic logic, the probabilistic part is an exact algorithm for belief propagation in Bayesian networks. The fuzzy-probabilistic calculus is created by YKY based on the Beta distribution.

Bootstrapping: Genifer will be written in its own language, which is a **logical-functional** programming language based on Genifer's logic and an existing functional programming language such as Clojure or Haskell.

To-do:

Ch 1 (Introduction) Explain the new ideas that I learned about the relationship between propositional logic and topological logic.

Ch 2 (Architecture) Explain AIXI, algorithmic complexity, Solomonoff induction, etc. Explain distributive architecture. New idea that bootstrapping is possible.

Ch 3 (KR) — ok —

Ch 4 (Logic) New logic of concept composition. Ideas about equational unification and concepts. Explain background notions, eg paradoxes.

Ch 5 (Z) Add new idea on the "Java-girl paradox", which is in draft paper.

Ch 8 (Inference) Copy and paste Bayesian inference and factor graph stuff from the Lisp code to here.

Ch 9 (Pattern recognition) Matrix technique on similarity.

Ch 11 (Learning) A lot of new material is in the slides.

Ch 12 (NL) New idea of semantic parsing. New diagrams from GUI.

Ch 13 (Memory) Explain hierarchical clustering idea, ontology.

Ch 14 (Planning) May need re-think.

Ch 18 (Implementation) Bootstrap Genifer in its own language.

Appendix A Recommend more books for AGI sub-areas. Especially math books.

Contents

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)

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 advantages 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:

$$p \vee q \wedge r.$$

Compare it with 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.)

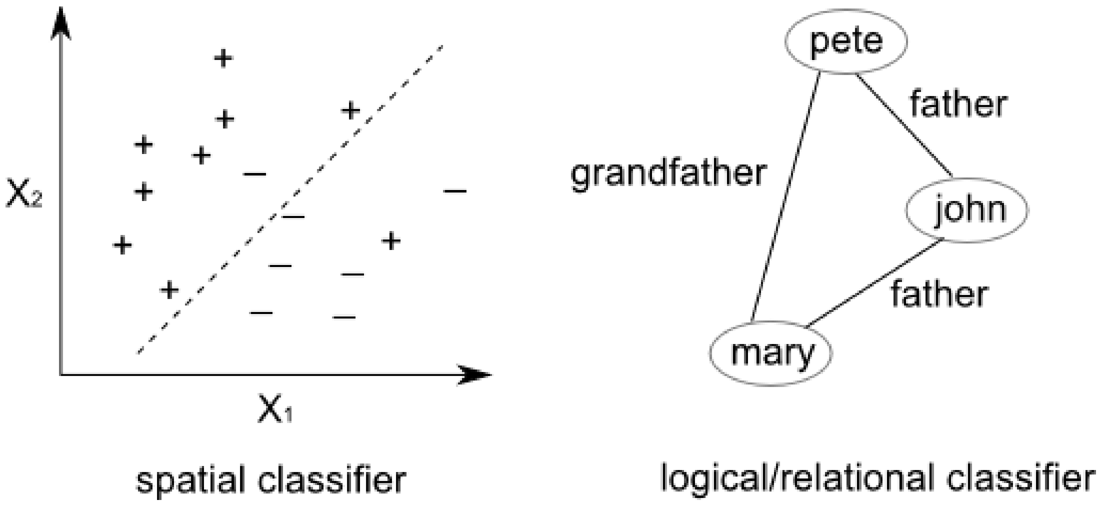
To put it more bluntly, I suspect that propositional approaches are rather useless in the logic-based AI framework (but I'd be pleasantly surprised to learn otherwise).

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.



spatial classifier

logical/relational classifier

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 "compressiveness" of predicate logic that I pointed out above. (After propositionalization, some fast propositional SATisfiability algorithms can

[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.

be invoked, but they are still qualitatively different from the spatial learning algorithms.) From this consideration, my current strategy is to focus on algorithms specifically for FOL / HOL / predicate logic.

The application of kernel methods to logic seems to rely on syntactic distance rather than *semantic* distance. (Eg: [Muggleton, Lodhi, Amini, and Sternberg, 2005] developed a support-vector ILP method; [Gartner, 2008] describes a method that constructs kernels for (first-order and higher-order) logic formulae, based on a representation of logic by typed lambda calculus in [Lloyd, 2003]. It involves the use of a "matching kernel" that measures syntactic distance only.) So far I have not seen an effective way to estimate semantic distance without performing logical inference.

{ 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], [Hammer and Hitzler, 2007]). }

{ To-do: I've gained some new insight into the issue of mapping predicate logic to continuous space, via algebraic logic. }

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

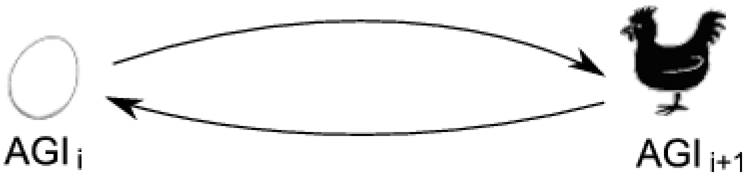


Figure 1.2: chicken and egg problem

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.

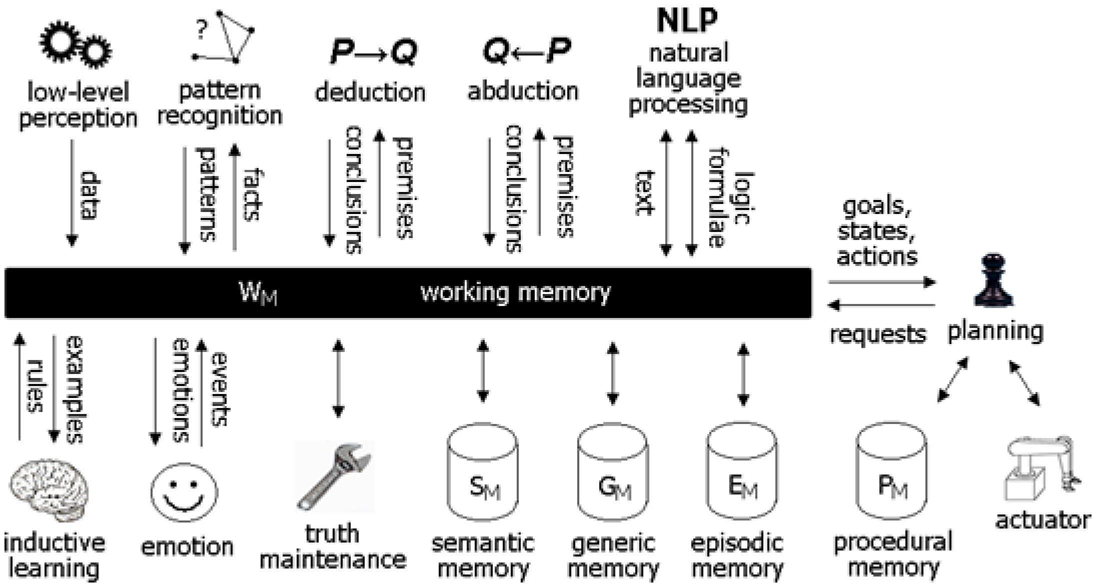
2 Architecture

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 §14.1, "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** (§9) which is the same as forward-chaining (§8.2). (NL input can bypass this step.) The result will be a set of *new facts* and will be put in Working Memory. Forward chaining can be performed several times on top of previous results.

(2) Perform **consistency check** (§10.3) on the new facts against the KB

(3) If a new fact is inconsistent with the current KB, invoke **conflict resolution** (§10.4)

(4) Perform **abduction** (§8.3) on the new facts, so WM will make appropriate assumptions to account for them.

(5) Invoke the **inductive learner** (§11.2), ie, try to compress the new facts + KB.

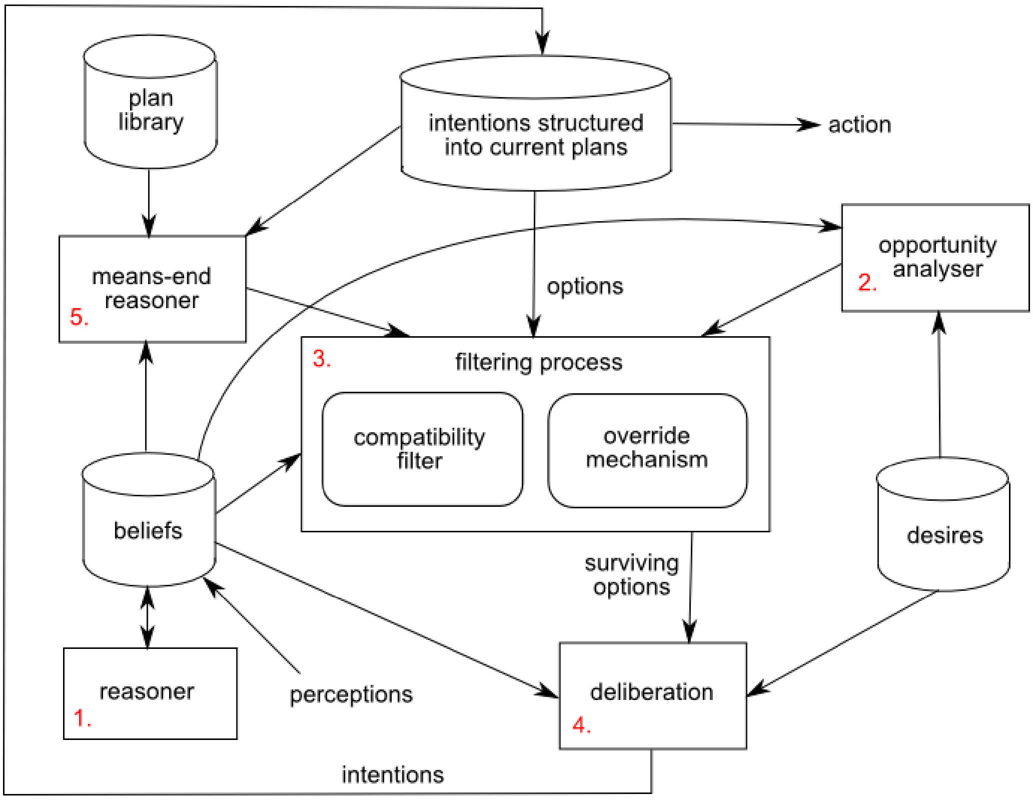


Figure 2.1: schematic diagram of the BDI architecture

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 §8.3), 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:

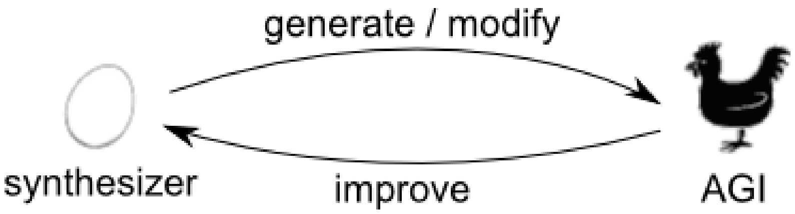
1. recognizes environmental states (eg *"the apple is on the table"*) from raw percepts (eg pixel values from a camera)
2. recognizes possible actions (eg *"I can put the apple on the plate"*)

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.

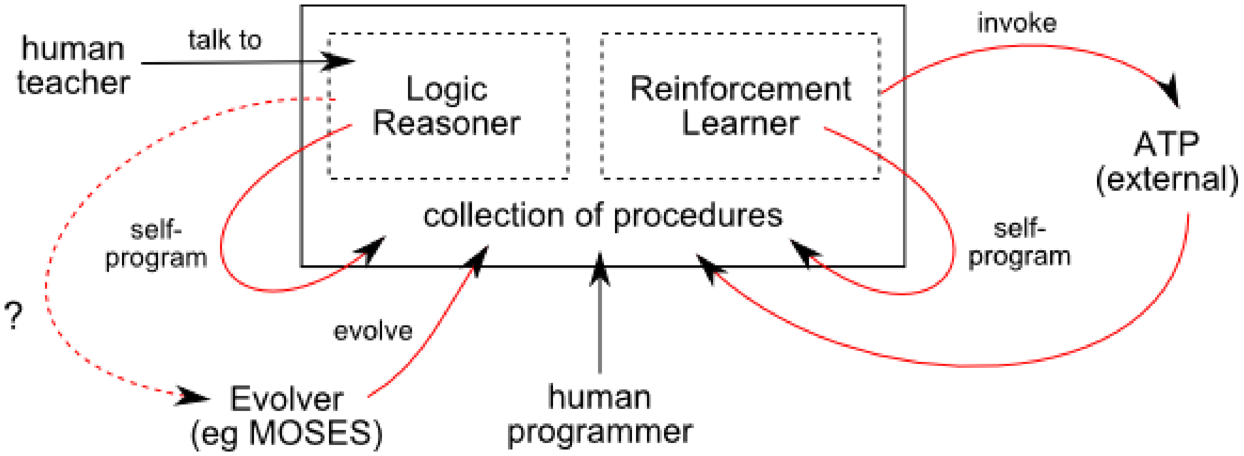
2. Natural reasoning (cf §1.6)
(a) The problem is that NR does not yet exist, but it is a *sine-qua-non desideratum* of AGI.
(b) So the question is how to enable an initially weak form of NR to synthesize programs. One possibility is to use **creativity**: the AGI will generate programs by partly reasoning 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** (§14.6).

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:



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

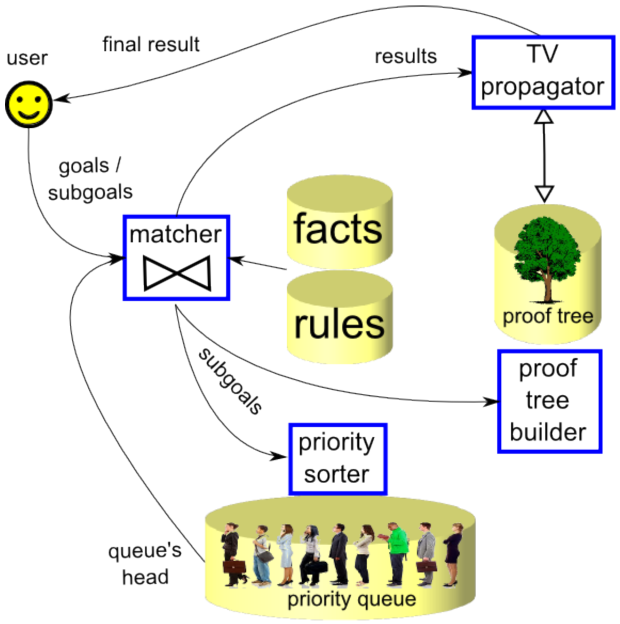
The question is to choose the *most effective* method of AGI self-programming (red lines). My intuition is that the evolutionary method is much less efficient than the other 3 (it is easier to explain to children to do something than to train animals by reinforcement, which is in turn easier than breeding 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 reasoning and deductive planning — for instance, to allow RL to invoke IE (inference engine) or vice versa.

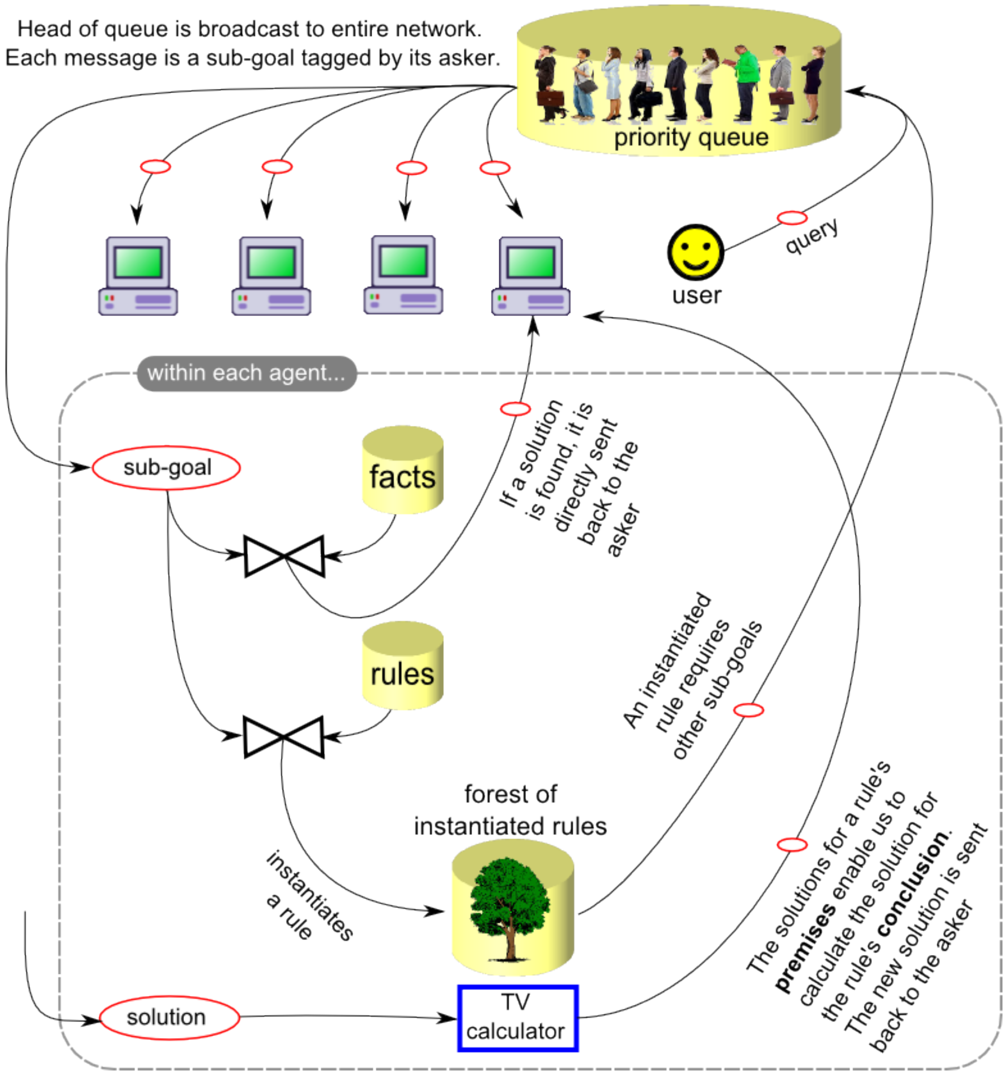
2.6 AIXI

2.7 Distributive architecture

This is a monolithic view of the inference engine. For simplicity's sake, only the deduction part is shown, since deduction is the most essential core of the engine.



This is a distributive architecture for deduction:


Head of queue is broadcast to entire network.
Each message is a sub-goal tagged by its asker.
priority queue
query
user
within each agent...
sub-goal
facts
If a solution is found, it is directly sent back to the asker
rules
An instantiated rule requires other sub-goals
forest of instantiated rules
instantiates a rule
The solutions for a rule's **premises** enable us to calculate the solution for the rule's **conclusion**. The new solution is sent back to the asker
solution
TV calculator

Each agent responds to queries and spits out solutions. For example, an agent may have a rule in its (local) KB that "professors who wear sandals are nice to students". The agent listens to queries, trying to find something it can answer. A query such as "who is nice to students?" would be a hit. Then the agent either:

1. returns an answer, if it knows as a *fact* that XYZ is nice to students.
2. applies a rule and returns one of more sub-goals; In this case, the sub-goal is "does XYZ wear sandals?" Wait for other agents to answer that.

In case #2, if another agent provides an answer "Professor Matt Mahoney wears sandals", say with TV = 0.9, and sends it back to the first agent, then the first agent decides how to calculate the TV of the conclusion given that TV of premise = 0.9. The only calculation it needs to perform is for the rule that it owns. Then it returns the answer to the asker.

This architecture is so wonderful because there is no need to construct the proof tree anymore. The proof tree seems to have disappeared but it is really implicitly constructed among the network of agents and the messages!

Thanks to Matt Mahoney for proposing the CMR (competitive message routing) architecture.

3 Knowledge representation

3.1 Introduction

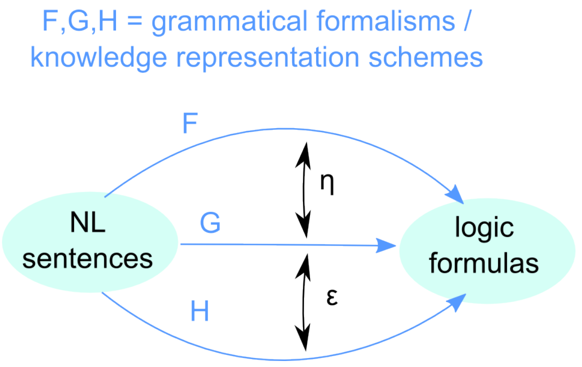
An AGI uses a KR structure to *represent* the external world, and this structure is built with limited computational resources, and as such, must be approximate. We have a lot of freedom to choose the KR format.

I chose logic as KR because it offers a *direct* way to translate natural-language texts into machine knowledge. This is of critical importance because the overall feasibility of AGI hinges on the efficiency of machine learning, and the most effective machine-learning method is "learn by being told" (§11.1).

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 as a *computational structure* for *constructing* mental models.

3.2 Multiplicity of knowledge representation schemes

This is how I think of the issue of knowledge representation:


F,G,H = grammatical formalisms /
knowledge representation schemes
F
η
NL
sentences
G
logic
formulas
H
ε

There are various KR schemes. For example, in NL, there are various grammar formulations:
F may be Phrase Structure Grammar
G may be Fluid Construction Grammar
H may be Categorial Grammar
J may be Genifer's machine-learned grammar (which may be stochastic and inscrutable to humans)
K may be some spatio-temporal KR scheme such as temporal logic, event calculus, situation calculus, etc

And then there would be transformations $(\eta, \epsilon, ...)$ between the grammatical formalisms.

The problem is whether we can mix multiple KR schemes like F and G. The commonsense KB would be different under either F or G, and the entire KB needs to be transformed by some η.

If we use $F, G, H, ...$ at the same time, confusion may arise during machine learning and reasoning.

Perhaps, if we make Genifer be aware of transformations like $\eta, \epsilon, ...$ then maybe it can deal with multiple KR schemes at the same time? Therefore we maybe don't need to choose or commit to any particular KR scheme at this stage... in other words, any would be fine.

3.3 Natural language

Please also see the chapter on natural language (§12).

3.3.1 Composition of concepts

In 1928 the young Haskell Curry started working on **combinatory logic** (CL) as a *logica universalis*, but it ran into inconsistency problems [1]. One special feature of CL is **combinatorial completeness** – meaning that any concept can, in principle, be applied to any other concept. One can readily see that, when the concept of *self-application* is applied to itself, it can lead to Russell's paradox (§4.8).

Nevertheless, combinatorial completeness is something desirable in a universal logic. It enables a logic to *reason about logical paradoxes themselves*. Bertrand Russell invented **type theory** to get around the problem of paradoxes, basically by banning all circular references. But as a result of that, type theory (and thus the higher-order logic built on top of it) cannot be used to reason about paradoxes.

Curry's combinatory logic can represent arbitrary combinations of concepts, in a manner such as:

$\lceil$wisdom$\rceil$ • $\lceil$socrates$\rceil$ = $\lceil$the wisdom of socrates$\rceil$

One of my latest insights is that the combination of concepts can be computed by the same process as **unification** of terms in logic. For more details, refer to §4.5.

In 1963 JA Robinson discovered resolution, which is really unification + propositional refutation. Unification decides if 2 terms can be made equationally identical. Propositional refutation is an inference step that deals with the "calculus of thinking" at the *sentence* level.

Around the 1900s, George Boole, CS Peirce, Gottlob Frege, amongst others, developed **predicate logic**, which is later popularized by such people as Hilbert, Ackermann, Russell, and Whitehead. Predacate logic has really been popular for only about 80 years, versus syllogistic logic that has been around since Aristotle's time. Predicate logic differs from propositional logic by giving propositions *internal structure* – a proposition is broken down into a **predicate** and one or more **objects**. However, this decomposition seems inadequate to deal with arbitrarily free combinations of concepts.

Combinatory logic provides a free way to compose terms via "application". The key is to regard terms as (compound) concepts. For example, in Genifer's notation:

"tall handsome guy"

is the combination of the concepts

(tall, handsome) ∘ guy.

Now, a few examples:
"tall handsome guy" is equivalent to *"handsome tall guy"*;
"very tall guy" implies *"tall guy"*; but
"very tall guy" does not equal *"tall very guy"*;

Thus the unification is modulo some special rules such as commutativity, associativity, etc, and certain reduction rules. In other words, unification modulo a theory = "the calculus of concepts".

[1] In Genifer we try to use fuzzy-probabilistic truth values to get around this problem (see §4.8).

So we have a neat decomposition:
calculus of thoughts = calculus of concepts + calculus of propositions

In **category theory**, one enforces the associative composition of arrows:
$(ab)c = a(bc)$. [2]
Likewise, I observed that if we enforce the associative composition of concepts, the logical form becomes very elegant. Due to associativity, any pair of concepts in an application:
$a \circ b$
must be *meaningful* in a meaningful sentence.

A consequence of enforcing associativity is that we can no longer use **Currying**. For example:
"John loves Mary"
would be rendered in Genifer's logic as
((mary loves) john)
rather than as suggested by Currying:
((loves mary) john)

Thus, in Genifer's logic, the 2 primitive operations are:

1. **application**, $\circ$
2. **pairing**, or cross product, (a, b)

with the associative and distributive rules:

$$(a \circ b) \circ c = a \circ (b \circ c)$$

$$(a, b) \circ c = (a \circ b, a \circ c)$$

One can recognize that this logic has the exact form as a category. To do: I'm curious as to the role of exponentiation in this category?

For more about how natural language sentences are translated into logical form, see the section on natural language and Geniform, §12.4.

Dynamic interpretation of semantics

Look at these examples:
glass slippers are made of glass
a door knob is part of a door
street prostitutes work on the streets.
However,
glass slippers do not work in glasses
a door knob is not made of doors
street prostitutes are not part of the streets.

This suggests that the **semantics** of the compound concepts depend on *external* pieces of knowledge, and hence must be interpreted *dynamically*. This is essentially the same idea as the abductive interpretation of natural language (§12.2).

3.3.2 Reification

Reification means "to turn into objects". For example:
"John loves Mary"
can be rendered as
loves(john, mary)
but
"John loves Mary deeply"

[2] We often omit the application symbol $\circ$.

would have to rendered as:

love($love_1$) ; $love_1$ is an instance of love
subject($love_1$, john)
object($love_1$, mary)
deep($love_1$).

Notice that in the reified form, the original base-level formula:

loves(john, mary)

is lost! This is rather unsatisfactory.

Geniform solves this problem by eliminating reification.

3.4 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 (§17), I think temporal representations can follow a similar scheme.

3.5 Contexts

Background: 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. Also the book [Bonzon, Cavalcanti, and Nossum, 2000].

In Genifer, a context can be regarded as a set of propositions that are *true in that context*. For example, a context would be the set of propositions:

{ *"It is late at night, on earth."*
"I am hungry."
"The fridge is empty."
"There is no money in my wallet." }

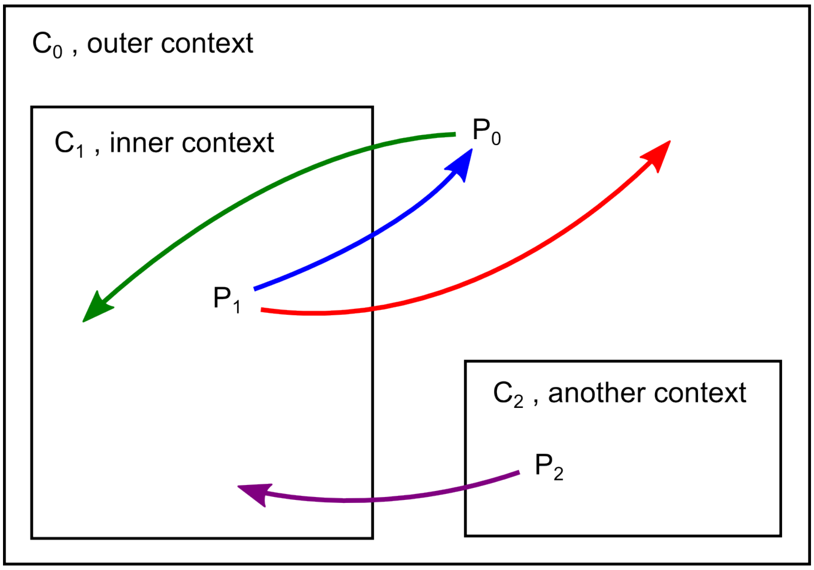
Such contexts are very useful in the clustering of the KB for inference speed-up (§**??**).

Some contexts are *partial* propositions, such as:

"In the kitchen..."
"John believes that..."
"As far as I'm concerned..."
"Theoretically speaking..."

In Genifer we calculate contexts as a component of the truth values of formulas. The reason we do so is:

3.5.1 Inference in contexts



In general there are 4 possible cases:

1. (Green) An assertion in an outer context, without contradiction, can apply to an inner context.
   For example:
   "In children's stories, animals can talk." (Inner context)
   "Cats are animals." (Truth in outer context)
   $\models$ *"In children's stories, cats can talk."*

2. (Red) An assertion in an inner context, even without contradiction, cannot apply to outer contexts.
   For example:
   "John believes he can fly." (Inner context)
   $\not\models$ *"John can fly."* (Outer context)

3. (Blue) When 2 assertions are in contradiction, the one in the **more-specific** context wins.
   For example (cf Example 4.6.1):
   "Mary does not know Kent is Superman." (Inner context)
   "Kent is Superman." (Outer context)
   "Superman can fly." (Outer context)
   $\therefore$ *"Kent can fly."* (Outer context)
   $\models$ *"Mary does not know Kent can fly."* (Inner context wins)

4. (Purple) When 2 contexts are **incomparable**, an assertion cannot apply from one context to the other.
   For example:
   "In John's mind Mary is dumb." (Inner context 1)
   $\not\models$ *"In Mary's mind Mary is dumb."* (Inner context 2)

These rules are stipulated only for the sake of facilitating compression; indeed, the ultimate purpose of using a formal logic is to facilitate algorithmic compression.

Remember Pei Wang's idea (§7) is to assign confidence to formulas that allows defeasible reasoning. Basically, for 2 formulas P, Q:

$$\begin{array}{ll} \text{IF} & \text{confidence}(P) > \text{confidence}(Q) \\ \text{THEN} & P \text{ overrides } Q. \end{array}$$

Our general rules can be summarized as follows. In context c:

$$\begin{array}{rll} P > \emptyset & \text{IF} & \text{context}(P) \supset \text{context}(\emptyset) \\ \emptyset > P & \text{IF} & \text{context}(\emptyset) \supset \text{context}(P) \\ P_A > P_B & \text{IF} & \text{context}(P_A) \subset \text{context}(P_B) \\ \emptyset > P & \text{IF} & \text{context}(\emptyset) \supset\!\subset \text{context}(P) \end{array} \tag{3.1}$$

where $>$ means "overrides / defeats", $\emptyset$ is the null formula, ie, the absence of any formula, context$(\emptyset) = c$ is the current context, and $\supset\!\subset$ means incomparable. Notice the reversal of sign in the 3rd rule, which is why I cannot compress all 4 rules into a single one.

A question is whether Pei Wang's NARS confidence can be merged with contexts — it seems that there are 4 independent truth values (P, Z, c, Cn): probability, fuzziness, NARS confidence, and context. Even if c and Cn can be merged, we still need to generalize confidence to **partial orders** so that $c(P)$ and $c(Q)$ may be incomparable.

3.5.2 Context management

Contexts are nodes of a global **Context Tree**, and can be encoded by numeric labels of the form $n_1.n_2.n_3...$ with $n_i \in \mathbb{Z}$. Now the question is how to assign contexts to formulas.

The problem seems similar to reference resolution in natural language processing, but is complicated by the need to manipulate contexts with logical rules (if we want to use logic to parse natural language).

3.6 Assumptions and counterfactuals

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

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

4 Logic

"The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate [calculemus], without further ado, to see who is right."

— Leibniz

Note: This chapter is currently very chaotic because of a re-organization effort. Fuzziness, $\mathcal{Z}$, is no longer considered a foundational part of the logic.

4.1 Background

4.1.1 Turing universality

Undecidability of FOL.

4.1.2 λ-calculus

(See Wikipedia: Lambda calculus).

Bound variables:

In the abstraction $(\lambda x.t)$ we call x the bound variable and t the body. Every occurrence of x in t is **bound** by the abstraction. Conversely, an occurrence of a variable y is **free** if it is not bound, eg in $(\lambda z.(\lambda x.(yx)))$.

Head-normal form:

A λ term is in head-normal form if, for $m \geq 0$ and $n \geq 0$, it can be expressed as:

$$\lambda x_1 ... x_m .\ x t_1 ... t_n .$$

The variable x may either be free or bound (one of $x_1, ..., x_m$).

De Bruijn indexes.

Director strings.

4.1.3 Functional programming

4.1.4 Combinatory logic

(See Wikipedia: Combinatory logic).

4.1.5 Term-rewriting systems

4.1.6 Simple type theory

4.1.7 Higher-order logic

HOL is roughly synonymous with type theory, with the addition of axioms that define logical primitives.

Standard semantics and general semantics (of higher-order logic).

How is Henkin semantics reducible to FOL?

4.1.8 Model theory

Model theory is the study of the truth of syntactic formulas defined via their correspondence to certain mathematical structures known as models. Originated by Alfred Tarski.

4.1.9 Proof theory

4.1.10 Deduction systems

Hilbert systems.

Sequent calculus.

Natural deduction.

Tableau.

Resolution.

4.2 Concept composition

4.2.1 Fragments

Is it possible to have inference at the sub-propositional level? Inference rules that operate upon fragments of propositions? Sometimes our human minds can focus on fragments or concepts that are not complete sentences.

For example, we may think of "spaghetti with meat balls" instead of "spaghetti with meat balls are tasty". Or we may think "To lose more weight, I need to..." without being able to complete the sentence immediately.

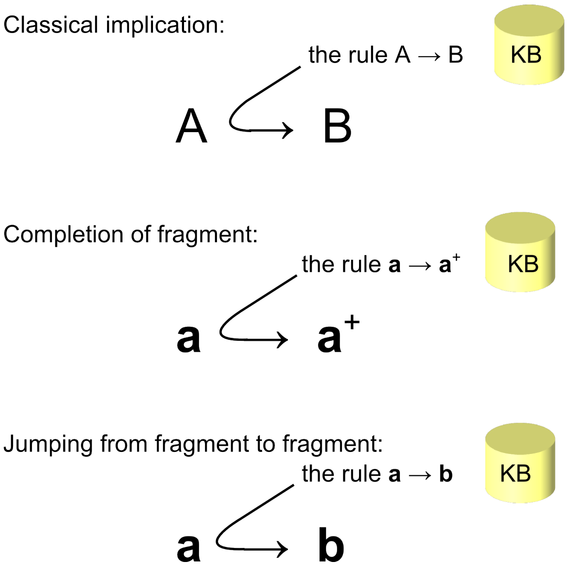
It seems possible to extend the idea of logical inference from propositions to fragments:

logic of propositions	**logic of fragments**
$A \rightarrow$ B (implication)	$\boldsymbol{a} \rightarrow \boldsymbol{b}$ (completion and jump)
special/general-ize in subsumption order	special/general-ize in ontology
$\approx$ (similarity of formulas)	$\approx$ (similarity of fragments)
$\neg A$ (not)	$\overline{\boldsymbol{a}}$ (opposite)

Completion means to extend a fragment to become a longer fragment or a complete proposition.

Jump means to jump from a fragment to another fragment or proposition in the KB.

Notice that completion and jump are supported by the KB, in the sense that they draw information from the KB in order to "push" through the arrow, just like in deduction. The following figure illustrates their similarity:



Note that in all 3 situations, the logic rules are supplied by the KB, which allows an "arrow" to be there in the first place. (The premises are also supplied by the KB, but is not shown in the figure.) The point is that the conclusions derived from fragments are not arbitrary; they depend on truths in the KB: a formula with the generalized arrow is a form of truth; and the premise fragment is also a form of truth – it can be interpreted as "Fragment $\boldsymbol{a}$ is worth paying attention to at this moment".

Now we can see the symmetry of the 3 forms and that $A \rightarrow B$ can be seen as a special case of $\boldsymbol{a} \rightarrow \boldsymbol{b}$ (jump). This leads to the unification of propositional and sub-propositional logic, with a unified "arrow". Thus, our logical operators are reduced to composition ($\circ$, product), union (sum), and exponentiation ($\rightarrow$, arrow). This satisfies the requirements to be a **CCC** (Cartesian closed category), and thus "feels right".

4.3 Uncertainty

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]:

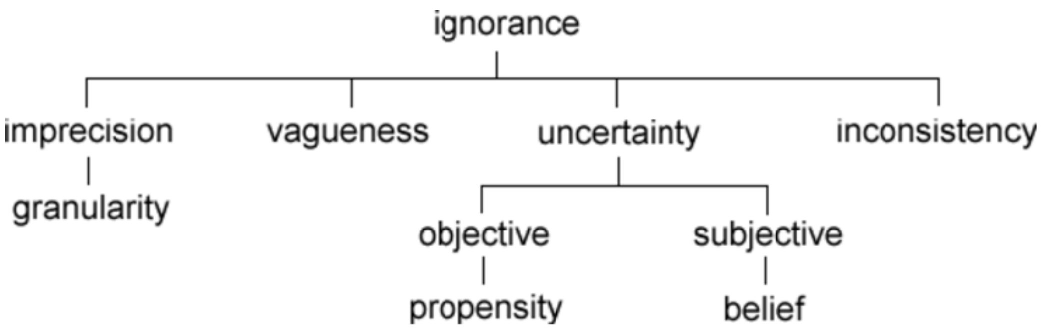


Figure 4.1: taxonomy of ignorance

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 §5.2 and 6.1.

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.

4.4 Nonmonotonicity and defeasible reasoning

One can create defeasible logics from classical logic, for example Reiter's **default logic** and McCarthy's **circumscription**. But the classical approaches seem to require enumerating all possible exceptions, which is impractical.

2 examples:

A.
1. John is usually very punctual.
2. Therefore, John will arrive at the airport on time.
3. John has an accident en route to the airport, and dies.
4. Therefore, John will NOT arrive on time.
Will John arrive on time?

B.
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 example B? 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 "it is sex without physical contact".

If we have a rule 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] in NARS provided a solution. His idea is not to store the exceptions to rules, but instead allow a *multitude* of rules to fire, calculate the "confidence" (§7.2) of each conclusion, and pick the conclusion with the highest confidence. This allows us to handle exceptions relatively easily.

For the example:

$$X \leftarrowtriangle A$$
$$X \leftarrowtriangle B$$

Pei Wang's idea is to calculate the combined conditional probability by weighing each individual conditional probability with their associated **confidences** (§7.2):

$$P(X|A,B) = \frac{P(X|A)c_A + P(X|B)c_B}{c_A + c_B}$$

which is a nice idea[1], but Abram Demski came up with an alternative idea that is more in accord with Bayesianism. The idea is to construct $P(X|A,B)$ from the marginal conditionals $P(X|A)$ and $P(X|B)$ and priors $P(A)$ and $P(B)$. This is achieved by applying Bayes' rule twice:

$$\begin{aligned} P(X|A,B) &= \frac{P(A,B|X)P(X)}{normalization} \\ &= \frac{P(A|X)P(B|X)P(X)}{normalization} \\ &= \frac{P(X|A)P(A)P(X|B)P(B)}{P(X)\ normalization} \end{aligned} \tag{4.1}$$

It may be possible to use Abram's method to construct all joint CPTs, instead of using contrived probabilistic formulations of AND and OR.

4.4.1 An example

"John is usually punctual, therefore, John will arrive at the airport on time."
"John died en route to the airport, therefore, John will NOT arrive on time."

punctual(john) $= (P) = \langle 0.8 \rangle$
punctual(john) $\rightarrowtriangle$ arrive(john) $= (A|P) = \langle 0.8 \rangle$
dead(john) $\rightarrowtriangle$ arrive(john) $= (A|D) = \langle 0.05 \rangle$

Abram's method:

$$(A|P,D) = \frac{(A|P)(P)(A|D)(D)}{(P,D)(A)^{2-1}}$$

[1] Except that NARS truth values do not conform to probability theory.

4.5 Unification

Unification is the process of making 2 terms identical via substitutions. For example, we can use a substitution:

$$\theta = \{john/X, mary/Y\}$$

to transform a term like this:

$$loves(X, Y) \xrightarrow{\theta} loves(john, mary).$$

Unification ($\bowtie$) is the process of finding the substitution given the initial and final terms. So:

$$loves(X, Y) \ \bowtie \ loves(john, mary) = \theta.$$

When we use logic to encode natural language and common-sense reasoning, terms are representations of (simple or compound) **concepts** (cf §3.3.1). Thus unification plays the role of deciding if 2 concepts are the same; in other words, the unification algorithm embodies the **calculus of concepts**.

Two important relations of concepts are:

1. Equality (=). For example: clark-kent = superman. Can be generalized to fuzzy equality (≈). For example: cat ≈ dog (cats are similar to dogs because they are both pet animals).
2. Inclusion (⊆), in classical AI also known as the "is-a" relation. For example: dog ⊆ animal.

The = relation affects unification because substitution of equals for equals is valid. We use a set of equations, called an equational **theory**, to define what are considered equal. Then we have so-called **unification modulo equational theory**, or **E-unification**. The algorithm for equational unification is called **narrowing**, see §4.5.2 below.

A complication is that = should be replaced by the more general ≈, and we need to modify the traditional algorithms to handle diminishing fuzzy truth values when ≈ is repeatedly applied.

4.5.1 Higher-order unification

The standard algorithm for higher-order unification was formulated by Huet in 1974.

Huet's algorithm

🚧

4.5.2 Equational unification / narrowing

Narrowing is **term rewriting** with **unification**. The matching process of rewriting is replaced by unification, where both the rewrite rule and the term to be rewritten can be instantiated. Some basic texts on narrowing are [Baader and Nipkow, 1998], [Snyder, 1991], [Prehofer, 1998], [Holldobler, 1989].

The equational theory E is replaced by a rewriting system R by **orienting** the equations in a specific direction.

The **one-step narrowing relation** $\overset{R}{\Rightarrow}$ is induced by a TRS (term rewriting system) R. We write $(t \overset{R}{\Rightarrow}^{\sigma} t')$ to denote a substitution σ that enables the term t to be rewritten as t' using one rewriting rule from R.

Example 4.5.1
To rewrite:
    $g(f(a, x))$
when given the equational theory
    $f(a, h(a)) = c$
we can make the substitution $\{h(a)/x\}$, yielding:
    $g(c)$.
This step makes the term "narrower", hence the name I guess.

Narrowing is a search process with high complexity because at each step, there are 2 kinds of choices to be made:

1. which rewrite rule in R to try; and
2. which position within t to try.

4.6 Equality

4.6.1 Morning star / evening star problem

A problem with equality is illustrated by the classic example “Morning Star / Evening Star”. The following is a similar example with its rendition in logic:

Example 4.6.1

“Clark Kent is Superman.”	clark-kent = superman
“Superman can fly.”	can-fly superman
“Mary does not know that Clark Kent is Superman.”	¬ knows(mary, ”clark-kent = superman”)
“Mary knows that Superman can fly.”	knows(mary, can-fly superman)
“Mary does not know that Clark Kent can fly.”	¬ knows(mary, ”can-fly clark-kent”)

A possible solution is to distinguish ”clark-kent” (in quotes, ie, the Clark Kent that Mary knows) and clark-kent (without quotes, ie, the real Clark Kent). This idea is similar to the so-called ”use / mention” distinction:

Use: Cheese is derived from milk.
Mention: “Cheese” is derived from the old-English word “cyse”.

The problem is in general due to the presence of “contexts” (cf §3.5). For example

- In the context of children’s tales, animals can talk.
- In the context of Star Wars, Darth Vader is Luke’s father.
- In the context of Mary’s beliefs, Superman and Kent are different persons.

4.7 Modal logic

> *Modality, si! Modal logic, no!*
> — John McCarthy

My view is to use predicates to represent modality instead of using modal logics. This view was first advocated by [McCarthy, 1997].

One argument for the use of modal logic arises from the previous Example 4.6.1:

	“Mary knows Superman can fly.”	knows(mary, can-fly superman)
	“Superman is Clark Kent.”	superman = clark-ken
*	*“Mary knows Clark Kent can fly.”*	knows(mary, can-fly clark-ken)

but Mary may not know that Superman is Clark Kent. This problem can be resolved as in the last section (§4.6).

4.8 Logical paradoxes

4.8.1 Russell’s or Liar’s paradox.

4.8.2 Curry’s paradox.

4.8.3 Skolem’s paradox.

4.8.4 Tarski’s paradox.

4.8.5 Non-axiomatizability.

"First order logic has an effective notion of proof which is complete w.r.t. the intended interpretation. This is the content of Godel's completeness theorem. As a result, the set of (Godel numbers of) universally valid first-order formulas is recursively enumerable." But "the set of second order validities is not arithmetically definable let alone recursively enumerable and hence that an effective and complete axiomatization of second-order validity is impossible" [**?**].

4.9 Shortcomings of the current logic

Binary vs $\mathcal{P}/\mathcal{Z}$. It may be desirable 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.

4.10 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" (§4.11).

4.11 Higher order logic

[Lloyd, 2003] developed inductive learning based on HOL, a **typed lambda calculus**, and also a logic programming language (Escher).

Lloyd uses a form of 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 }

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.

🚧

4.12 Algebraic logic

My motivation for studying algebraic logic is to develop better inference and learning algorithms for logic. In the 1990s, the invention of SVMs by Vapnik revolutionized the field of statistical learning, and SVMs remain to be

[2] Ie, binary logic makes programming easier

the best general spatial learner today. What if we can invent something similar to logic as what SVM is to spatial domains?

4.12.1 Survey of algebraic logic

[Halmos, 1962], [Andreka, Nemeti, and Sain, 2001], [Halmos, 1988], [Craig, 1974], [Plotkin, 1994], [Lawvere and Rosebrugh, 2003]'s Appendix A.

To do: Lawvere's idea that the existential and universal quantifiers are adjuncts to each other. What use is it?

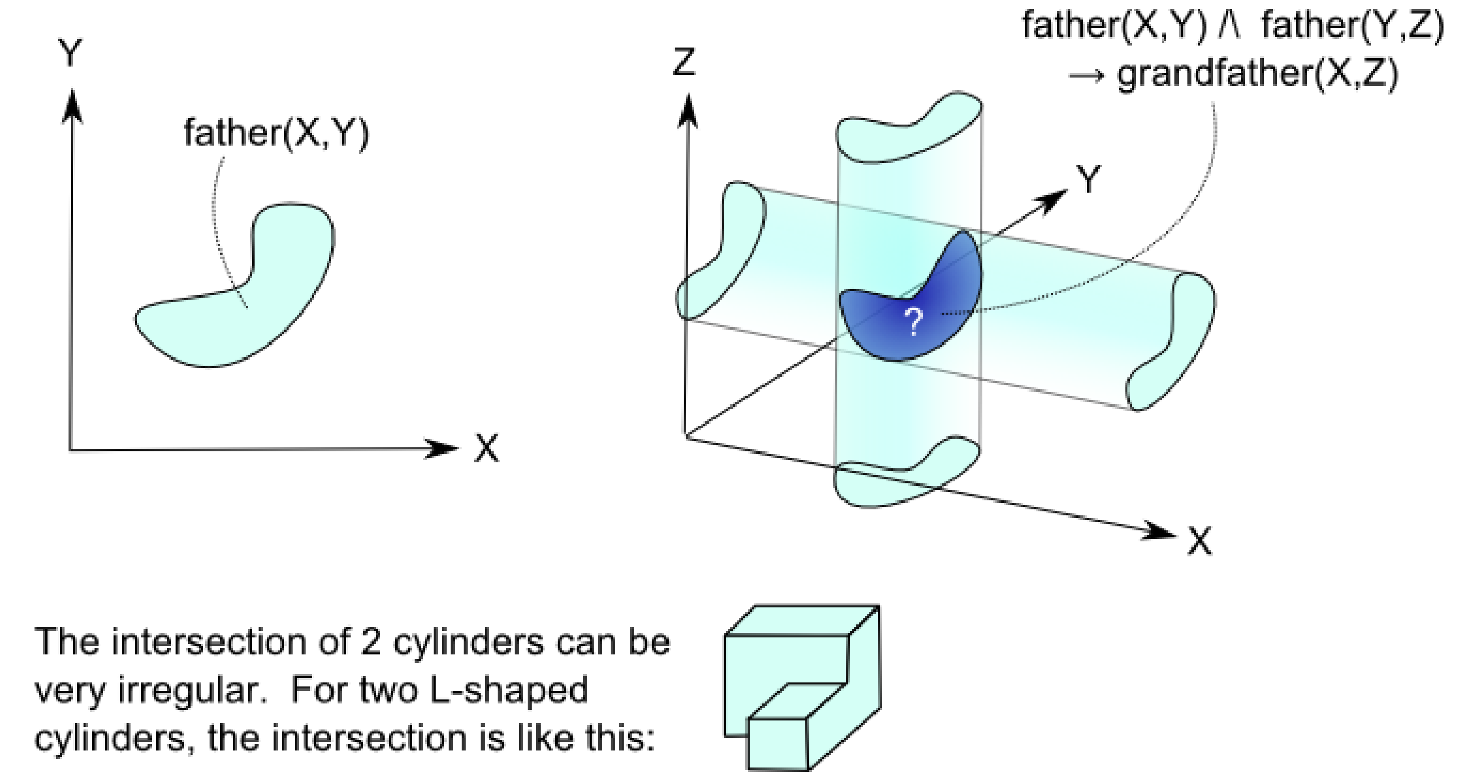
4.12.2 Speeding up inference and learning

For all logics that I know of, deduction is performed by *iterating* deductive steps. Each step is the application of an inference rule. If a formula with quantification is involved, some kind of pattern matching or unification is required apply the formula. Such an iterative process is equivalent to searching in a large proof space.

My idea is: perhaps we can "map" logic to some *continuous* space, and somehow approximate the proof process by spatial computational techniques?

One natural idea is:

first-order objects	$\longrightarrow$	points in space
predicates	$\longrightarrow$	regions / shapes in space
logical entailment	$\longrightarrow$	finding intersections of regions



The intersection of 2 cylinders can be very irregular. For two L-shaped cylinders, the intersection is like this:

We know that, for some concepts (predicates), the shape of the regions defining the predicate can be extremely irregular or even incomputable.

Our motivation is twofold:
1. How can we find intersections of regions efficiently?
2. Can we altogether avoid performing deduction via iterative steps?

One idea is to have multiple "domain grids". We'd be sorting the source domains into various clusters. That will speeed up the look-up of intersections.

However, this idea only gives us a geometric interpretation, but it doesn't tell us how to compute the shapes of regions efficiently. The only way to compute the regions is to go back to the basic inference rules of the logic, ie, iteratively and syntactically.

Another idea is the "dual":

first-order objects	⟶	shapes / regions in space
predicates	⟶	points in space
logical entailment	⟶	?

5 Vagueness ($\mathcal{Z}$)

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

NOTE: this section will be re-formulated soon. In the new version I only make 2 assumptions:

1. There are some quantities known as "degrees" and they are normalized to [0,1].
2. There exist relations amongst these quantities captured by **statistical models**.

I think the new formulation will be fairly unblemishable and uncontroversial, as opposed to earlier forms of fuzzy logic which many people are skeptical of. But the new version also falls out of the classification as logic and is more of a fuzzy-probabilistic calculus.

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.

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.

5.1 Vague phenomena

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

The first type is **discrete** in nature, for example[1]:
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

[1] I will later argue that even these concepts are not completely binary.

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[2]
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]).

5.2 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 knowledge. common-sense reasoning and natural language understanding depend on the use of vague concepts.[3] 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.)

5.3 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 §4.4 on nonmonotonic reasoning.

Having numerical vagueness allows us to:

1. Represent quantitative rules. For example:

[2] A solution to the sorites paradox is given in [Bergmann, 2008] p268-282, using fuzzy logic and the notion of *decaying validity*.

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

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* (§4.4). 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 $\mathcal{Z}$.

5.4 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].

$\mathcal{Z}$ 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**"[4]) 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?" [5]

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

[4] The term is used informally, not referring to Guha's notion of micro-theories in Cyc.
[5] Abram Demski suggested this to me in discussion.

just as H can be inexact, but this inexactitude is modeled separately by *distributing probabilities over* $\mathcal{Z}$ (§5.15).

$\mathcal{Z}$ is not an approximation of probability; §5.5 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 (§5.9)
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.

5.5 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

There seems to be two problems with this interpretation. The first concerns the min-max calculus (§5.8) – 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 §5.15.1 we will consider probability distributions over fuzziness, where this problem reveals a subtle difference in the shapes of probability distributions.

{ TO-DO: [Brachman and Levesque, 2004] proposed a probabilistic way to handling fuzziness which at a deep level is identical to my approach. }

5.6 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).

5.7 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.

$$Z_1(x) = e^{-\ln 2 \,\cdot\, (x/\xi)^2} \quad \text{and} \quad Z_2(x) = 1 - e^{-\ln 2 \,\cdot\, (x/\xi)^2} \tag{5.1}$$

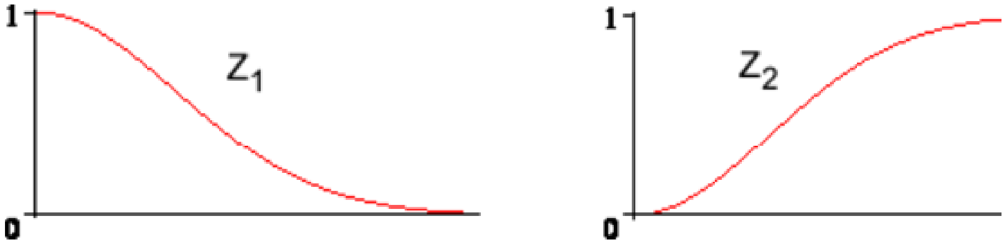


Figure 5.1: membership functions

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 §5.15.1.

The interpretation of the parameter ξ in eqn (5.1) 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":

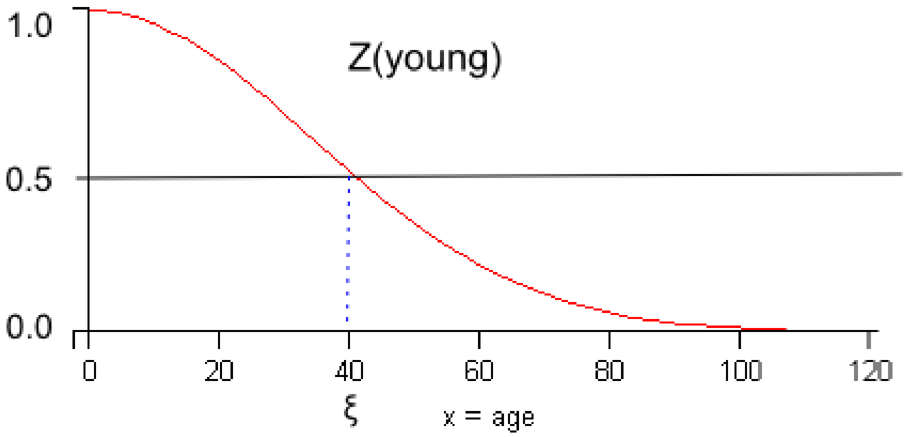


Figure 5.2: neutral point

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

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

Table 5.1

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.

5.8 Why Z obeys min-max calculus

Probabilities obey sum-product calculus; $\mathcal{Z}$ obeys **min-max calculus** [Zadeh, 1965], ie:

$$z_1 \overset{Z}{\wedge} z_2 \overset{def}{=} min(z_1, z_2) \tag{5.2}$$

$$z_1 \overset{Z}{\vee} z_2 \overset{def}{=} max(z_1, z_2). \tag{5.3}$$

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.

5.9 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.

5.10 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

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

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

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

All these are reasonable conclusions.

5.11 Unifying AND and OR

We seek to unify AND and OR by using a single operator $\odot$ with a parameter θ such that when $\theta = 0$ it reduces to AND and when $\theta = 1$ it becomes OR.

A simple way to define $\odot$ is:

$$X_1 \underset{\theta}{\bigodot} X_2 = (1 - \theta)(X_1 \overset{Z}{\wedge} X_2) + \theta(X_1 \overset{Z}{\vee} X_2). \tag{5.4}$$

The animation (Figure 5.3) shows the graph of $X_1 \odot X_2$ as θ varies.

(** Click to play **)

Figure 5.3: Graph of $X_1 \odot X_2$ as θ varies.
You need to have a PDF viewer that supports Flash, such as Adobe Reader.
Make the Reader trust this document.
For maximum pleasure click the "play" and "loop" options.

5.12 "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 **sfeatures** 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 $\overset{Z}{\wedge}$ 4-legged	1.0	typical old-fashioned chair
rotating $\overset{Z}{\wedge}$ has wheels	1.0	typical office chair
has wheels $\overset{Z}{\wedge}$ 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)
$z_1 \mathbin{\widetilde{\wedge}} z_2 = \dfrac{z_1(1-z_1)+z_2(1-z_2)}{1-z_1+1-z_2}$	$z_1 \mathbin{\widetilde{\vee}} z_2 = \dfrac{z_1 z_1 + z_2 z_2}{z_1+z_2}$

(5.5)

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.

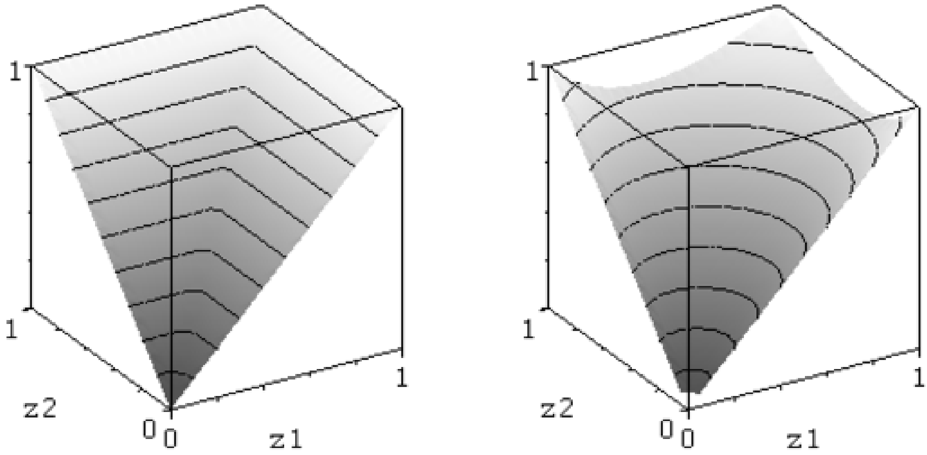


Figure 5.4: comparison of max and soft max

5.13 $\mathcal{Z}$ modifiers

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

extremely very moderately slightly

so we can express things like:

lukewarm ← moderately(warm)
obese ← very(fat)

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

$$z_0 := \Gamma(z_1) \tag{5.6}$$

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

$$z_0 := \Gamma(z_1; z^*) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z_1 - z^*)^2/2\sigma^2} \tag{5.7}$$

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

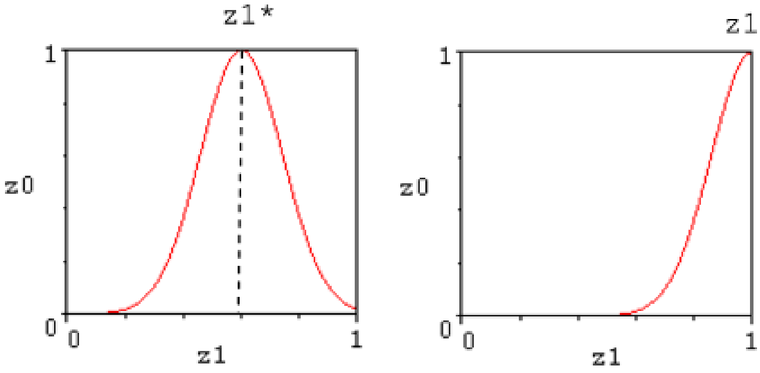


Figure 5.5: Fuzzy modifiers with $z^* = 0.6, 1.0$

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 ← $\Gamma_{0.6}(\text{warm}) \overset{Z}{\wedge} \Gamma_{0.8}(\text{warm})$

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.

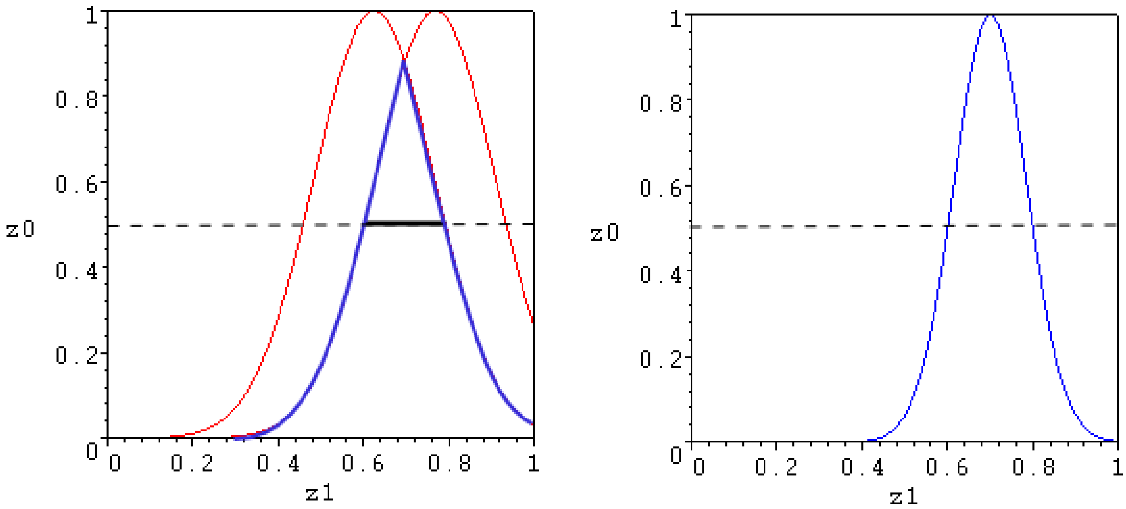


Figure 5.6: Two representations of "lukewarm"

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 $\overset{Z}{\wedge}$, $\overset{Z}{\vee}$ can be universal approximators. }

5.14 $\mathcal{Z}$-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):

$$z_0 := \overset{Z}{\bigvee_i} \overset{Z}{\bigwedge_j} \Gamma(z_{ij}) \tag{5.8}$$

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:

5.14.1 Traditional fuzzy logic

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

⇒	0	1
0	1	1
1	0	1

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

$$((z_1 \overset{Z}{\wedge} z_2) \Rightarrow z_0) = z_c$$

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 §6.5: 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.[6]

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

5.15.1 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:

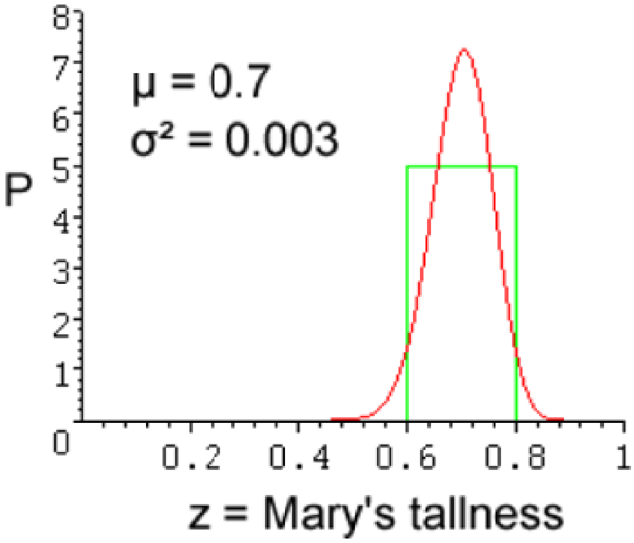


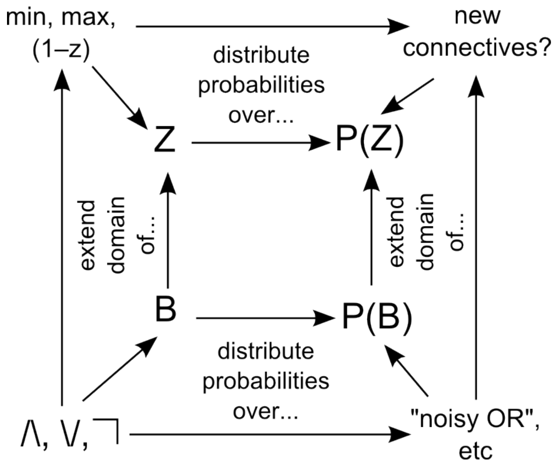
Figure 5.7: an example $\mathcal{P}(\mathcal{Z})$ distribution

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). §6.7 tried to justify this.

The following commutative diagram shows the relationship between $\mathcal{B}$, $\mathcal{P}(\mathcal{B})$, $\mathcal{Z}$, and $\mathcal{P}(\mathcal{Z})$:

[6] 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.



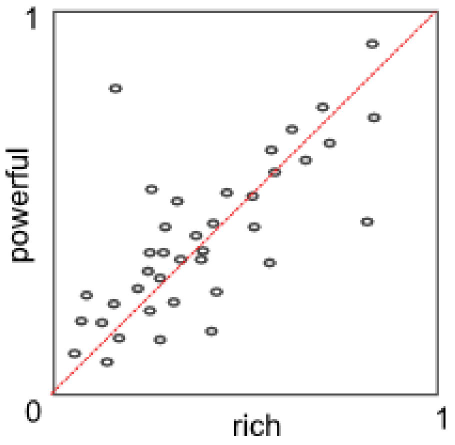
The inner square contains the 4 types of truth values. The outer square are the corresponding logics, showing their logical operators. The radial arrows point to the "underlying sets" of TVs.

5.15.2 An example

Consider the common sense notion:

The richer a person, the more powerful s/he is.

Note that this rule is inexact: there are exceptions (where some rich people are not powerful and some powerful people are not rich) and deviations (the data points do not fall on a straight line):



There are reasons to believe that the identity function (red line) represents the least-error regression function: first, the $\mathcal{Z}$ values are in "natural scale" (ie, $z = 1$ corresponds to ∞) so (z_1, z_2) at $(1, 1)$ is obvious if we accept the original statement; secondly, when $z_1 = 0.5$, the person is neither rich nor poor, and we should have nothing to say whether the person is powerful or not, as far as the original statement is concerned.

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

Up to now there are 4 possible TV types:

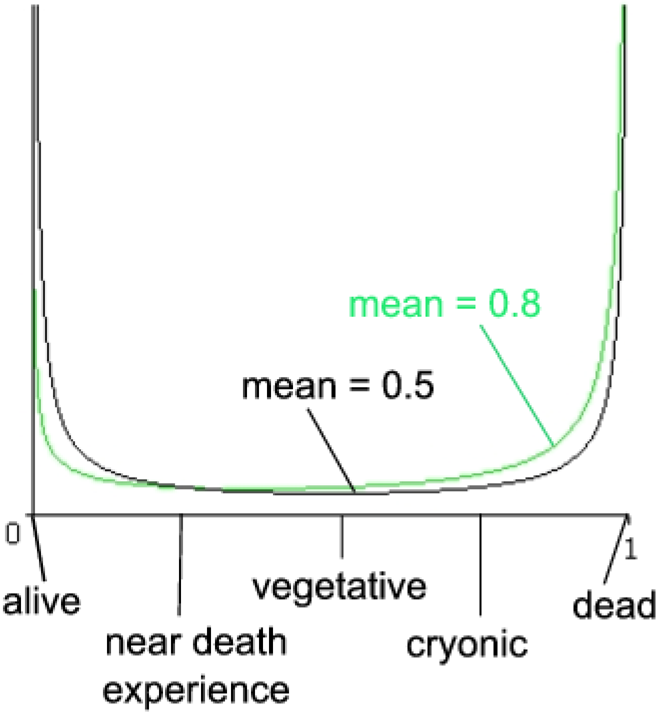
Table 5.2

truth values		
category	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 $\mathcal{B}$

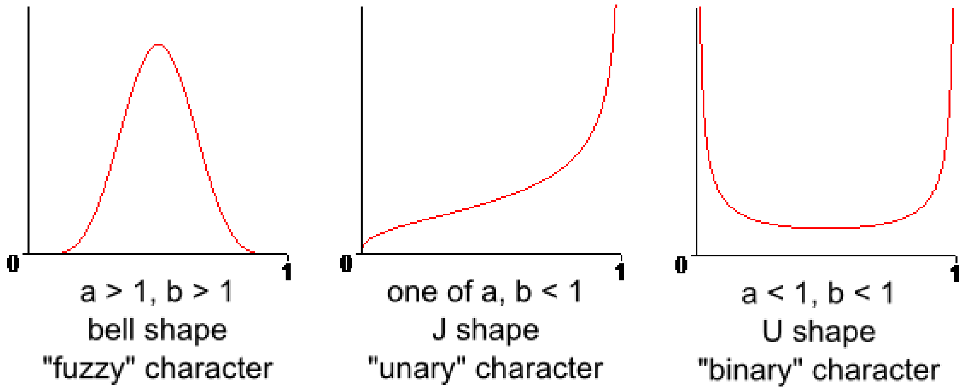
Some variables have a strong "binary flavor". For example, in common sense, a person is either dead or alive, although a more nuanced view will have grades of being dead. If deadness is a $\mathcal{Z}$ variable, z = 0 would be "definitely alive", z = 1 would be "definitely dead" (eg reduced to ash), 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 6.1) 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 $P(married) = 0.8, \qquad P(\neg married) = 0.2.$

In general, for any $\mathcal{P}(\mathcal{Z})$ variable Z, especially in the binary regime, one can set the probability of "Z" (viewed as a binary event) to be equal to the probability mass for $z \geq \frac{1}{2}$ which is given by the CDF of the Beta distribution, ie, 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:

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

Thus

$$P(Z = \text{true}) = P(z \geq \frac{1}{2}) = 1 - P(\overset{Z}{\neg}\, Z) = 1 - P(z < \frac{1}{2}) = 1 - I_x(1/2; a, b)$$

The desired mean μ can be solved from:

$$I_x(1/2; a, b) = p \tag{5.9}$$
$$\mu = a/(a + b) \tag{5.10}$$

An interesting observation: As the variance v increases, it eventually reaches a maximum v_{max} where $a = b = 0$ and the Beta distribution is undefined. From Mathematica experiments, if we keep $\mu = a/(a + b)$ fixed:

$$\lim_{a,b\to 0} I_x(1/2) = \mu - 1$$

This implies that, in the extreme polarized case:

$$p = 1 - I_x(1/2) = 1 - (1 - \mu) = \mu$$

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 §8.

6 Probabilities ($\mathcal{P}$)

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

6.1 Why $\mathcal{P}$ 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.

6.2 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:

$$f(x; a, b) = \frac{x^{a-1}(1-x)^{b-1}}{B(a, b)} \tag{6.1}$$

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:

$$\mu = \frac{a}{a+b} \tag{6.2}$$

$$v = \frac{ab}{(a+b)^2(a+b+1)} \tag{6.3}$$

and their inverse can be solved (by Maple) to give:

$$a = \frac{m(v+m^2-m)}{v} \tag{6.4}$$

$$b = \frac{(v+m^2-m)(m-1)}{v} \tag{6.5}$$

From the above it can be seen that the maximum variance occurs at:

$$v_{max} = m - m^2$$

at which point the probability mass is entirely at the poles.

Further discussion about how the Beta distribution represents $\mathcal{P}(\mathcal{Z})$ variables is in §5.15.3.

6.3 Causality

Three recent books that tackle the problem of causality from an AI perspective are: [Shafer, 1996], [Pearl, 2000], and [Williamson, 2005]. There is also a comprehensive reference [**?**].

As [**?**], Ch 21, explains: We know that a Bayesian network is directed, but the direction of the arrows do not have to be meaningful. They can even be anti-temporal. On the other hand, it is common wisdom that a "good" BN structure should correspond to causality, in that an edge $X \to Y$ often suggests that X "causes" Y, either directly or indirectly. Bayesian networks with a causal structure tends to be sparser and more natural. However, as long as the network structure is capable of representing the underlying joint distribution correctly, the answers that we obtain to probabilistic queries are the same, regardless of whether the network structure is causal or not.

It seems that causal relations cannot be captured simply by probabilistic relations but require some form of inductive algorithm to obtain, such as the IC (for "inductive causality") algorithm proposed by [Pearl, 2000].

6.4 Predicate 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*.

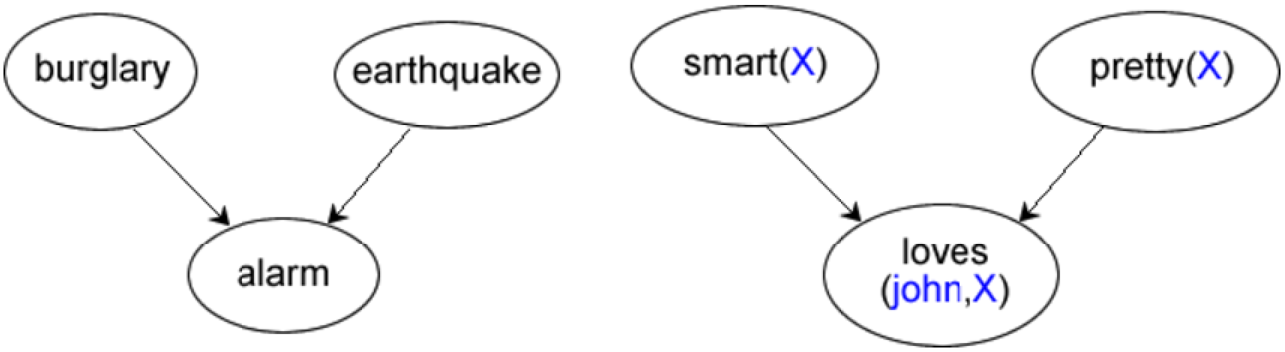


Figure 6.1: propositional vs first-order Bayesian networks

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...

6.5 Classical logic must give way to Bayesian logic

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

6.5.1 Classical implication is "out"

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

$$A \rightarrow B \equiv \neg A \vee B$$

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 networks. (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 the same one adopted by BLP, MEBN, BLOG, and PRM (see §6.4).

6.5.2 The probabilistic quantifier "#"

In addition to the classical quantifiers ∀ ("for all") and ∃ ("exists"), we can introduce a probabilistic quantifier # ("for some"). It returns as truth value the probability of the quantified statement.

For example, the truth value of:

#x. tall(x) "Some x's are tall"

can be defined as:

$$\frac{|\{x|tall(x)\}|}{|\text{reference class of } x|}$$

Notice that the reference class of x cannot be read from the formula above; it must be specified by the **type** of x, such as:

#x:human. tall(x).

How can the computer know the sizes of these sets? It seems that it has to keep a record of the sizes every time a new fact is encountered. For example:
When tall(john:human) enters the sensory stream, we should note that john belongs to the classes chinese, male, human, organism, etc. Each of these reference classes remembers its average tallness (stored as a distribution with its mean and variance, and a confidence, which gives the size of its support). These records will be updated according as John's tallness compares to the distributions.

In other words, for each property (ie, predicate) we have to record the distributions for (at least) the *typical classes that the predicate applies to*.

6.6 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 → miserable
¬ traitor → 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 §8.2.2.

6.7 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 §6.6 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.*[1]

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.

[1] We still need inference algorithms that can handle intervals as demonstrated in §6.6, but the intervals will be instantly converted to point-values after each step. This reduces complexity greatly.

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:

> "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.

6.8 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 law[2]):

		noisy AND	noisy OR
X_1	X_2	$X_1 \overset{P}{\wedge} X_2$	$X_1 \overset{P}{\vee} X_2$
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. }

Anyway, I defined my version of "probabilistic" AND and OR so that they obey 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.

It is better to write the complement $(1 - c)$ as superscript. So the pair $c_l, (1 - c_l)$ is written as $\vec{c}^{\,i} = c_i^0, c_i^1$. In this

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

notation my definitions can be expressed simply as:

$$X_1 \overset{P}{\wedge} X_2 = \sum_{ij} (\vec{c}_1^{\,i}\, \vec{c}_2^{\,j}\, \vec{x}_1^{\,i}\, \vec{x}_2^{\,j}) \tag{6.6}$$

$$X_1 \overset{P}{\vee} X_2 = \sum_{ij} ((\vec{c}_1^{\,i} + \vec{c}_2^{\,j} - \vec{c}_1^{\,i}\, \vec{c}_2^{\,j})\, \vec{x}_1^{\,i}\, \vec{x}_2^{\,j}) \tag{6.7}$$

which can be easily generalized to $n > 2$. When expanded, they yield the following table:

		probabilistic AND	**probabilistic OR**
X_1	X_2	$X_1 \overset{P}{\wedge} X_2$	$X_1 \overset{P}{\vee} X_2$
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	$c_1\ c_2$	$c_1 + c_2 - c_1 c_2$

(6.8)

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

$$X_0 := \overset{P}{\bigvee_i} \overset{P}{\bigwedge_j} X_{ij} \tag{6.9}$$

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:

$$X_0 := \overset{P}{\bigvee_i} \overset{P}{\bigwedge_j} X_{ij}; c_{ij} = \overset{P}{\bigvee} \{X_{11}; c_{11} \overset{P}{\wedge} X_{12}; c_{12} \overset{P}{\wedge} \cdots, X_{21}; c_{21} \overset{P}{\wedge} X_{22}; c_{22} \overset{P}{\wedge} \cdots, \cdots\} \tag{6.10}$$

It can be verified that *association* holds as in classical logic, ie:

$$\begin{aligned} (A \overset{P}{\wedge} B) \overset{P}{\wedge} C &= A \overset{P}{\wedge} (B \overset{P}{\wedge} C), \\ (A \overset{P}{\vee} B) \overset{P}{\vee} C &= A \overset{P}{\vee} (B \overset{P}{\vee} C), \end{aligned} \tag{6.11}$$

a fact that can be exploited for efficient calculation. This is because the CPT (conditional probability table) has size 2^n where n is the number of variables. But when the operator is associative, we can calculate $\overset{Z}{\bigwedge} X_i$ as $((X_1 \overset{Z}{\wedge} X_2) \overset{Z}{\wedge} X_3) \overset{Z}{\wedge}$ ..., without having to generate the full CPT. Thus the space complexity remains constant.

It is also possible to use the trick in §5.11 to combine AND and OR with the parameter θ:

$$X_1 \odot X_2 = (1-\theta)(X_1 \overset{P}{\wedge} X_2) + \theta(X_1 \overset{P}{\vee} X_2) \tag{6.12}$$

7 Confidence ($\mathcal{C}$)

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)).

7.1 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.

7.2 Confidence

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

$$w = w^+ + w^- \tag{7.1}$$

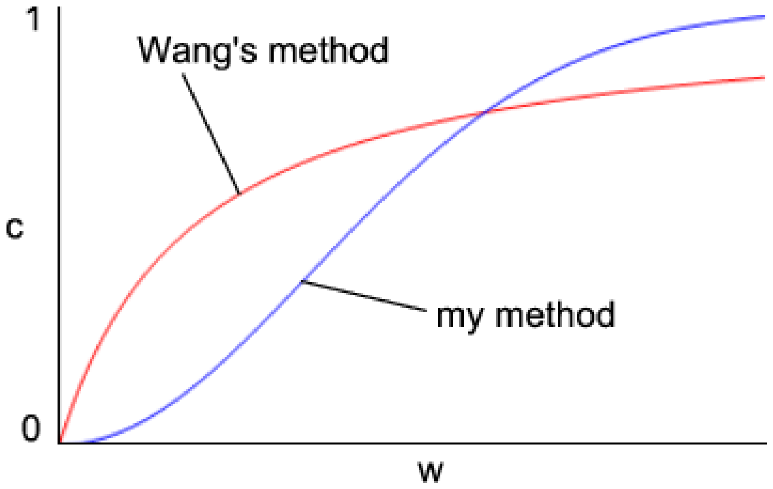
which is an integer from 0 to ∞.

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:

$$c = \frac{1}{1 + k/w} \qquad \text{(Wang's method)} \tag{7.2}$$

or

$$c = 1 - e^{-ln2 \cdot (w/k)^2} \qquad \text{(my method)} \tag{7.3}$$



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 §8.2.4).

7.3 Confidence and probability

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

$$p = f = \frac{w^+}{w} \tag{7.4}$$

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 §5.15.1 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.

7.4 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) → long-hair(X)
is supported by a number of instances such as:
positive example: female(mary), long-hair(mary)
negative example: female(jane), ¬ long-hair(jane)
The support of the rule is the *total* number of such instances that have been encountered[1].

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 §8.2.4.

[1] 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 }

8 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?

8.1 Some background

8.1.1 Resolution

In essence, the resolution rule is:

$$\frac{\begin{array}{l} P \vee Q \\ \neg P \vee R \end{array}}{Q \vee R}$$

Resolution is a popular deduction technique for binary logic; we will not use it but it is helpful to know.

8.1.2 Horn clauses and Prolog

In essence, a Horn clause is a rule of the form:

(Prolog notation) `A :- B, C, D, ...`

(logic notation) $A \leftarrow B, C, D, ...$

The Horn form makes deduction particularly efficient because it is like IF-THEN rules in production systems. Resolution in Horn form is the basis of Prolog.

8.1.3 Forward- and backward- chaining

These are deduction algorithms.

Forward-chaining *starts with the premises*, and applies deduction steps in the forward direction, to try to arrive at conclusion(s). A goal (conclusion) may be specified as the search termination criterion; if not, it is called goal-less forward-chaining.

Backward-chaining *starts with the goal to be proven*, and applies deduction steps in the backward direction. The search graph terminates with nodes that are the premises.

8.1.4 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.

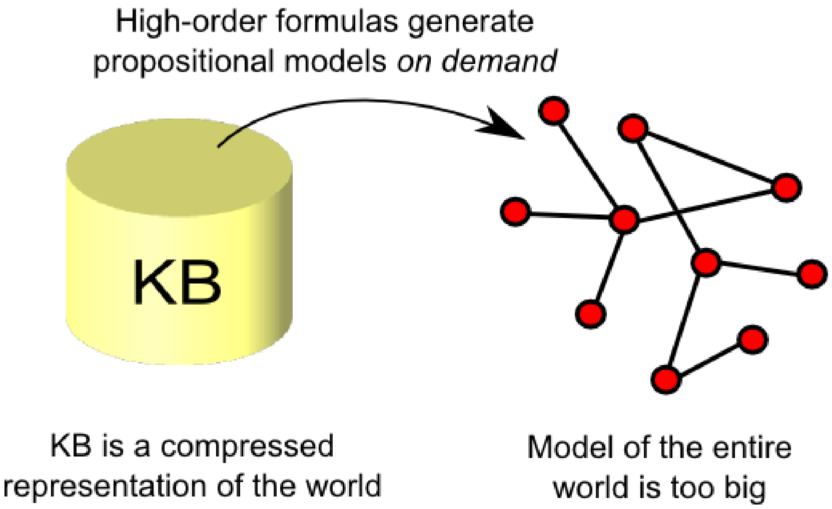
6. Pearl's algorithm, when operating on trees, is linear time. But trees seem to be insufficient for AGI reasoning.

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.

8.2 Deduction

8.2.1 Predicate logic and substitution management

We have argued that propositional (0th-order) logic is inadequate for AGI. The situation is illustrated as follows:


High-order formulas generate
propositional models *on demand*
KB
KB is a compressed
representation of the world
Model of the entire
world is too big

This is known as KBMC (knowledge-based model construction). What it means is that we need to **fetch** logical rules from the KB and **instantiate** those rules to form the propositional Bayesian networks upon which probabilities are propagated.

Substitution management in first-order logic provers is quite complicated. This is famously explained in classics such as SICP [Abelson, Sussman, and Sussman, 1984], where the authors advocated using **lazy sequences**, also known as **streams**.

Think of a sequence of **solutions** attached to each variable node of the proof tree. Each solution consists of a **substitution** and a **TV**. When we carry out an operation between 2 variables, say $X^1 \odot X^2$, we are not just dealing with 2 objects, but with 2 *sequences* of possible solutions. Let's denote the sequences as $X^1 = \{x^1_1, x^1_2, ...\}$ and $X^2 = \{x^2_1, x^2_2, ...\}$. For each solution of each sequence, we need to:

[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.

1. Match the 2 substitutions (x_i^1.sub and x_j^2.sub), ie, see if they are compatible, and if so, merge the 2 substitutions. Let's denote the matching operation as (x_i^1.sub $\bowtie$ x_j^2.sub).
2. If the substitutions are compatible, perform the calculation of the CPT (conditional probability table). Let's denote this operation as (x_i^1.TV $\odot$ x_j^2.TV).

Now denote the combined action of $\bowtie(\cdot,\cdot)$ and $\odot(\cdot,\cdot)$ as $\heartsuit(x_i^1, x_j^2)$. Then what we need to calculate can be expressed as:

$$X^0 = \heartsuit(X^1 \times X^2 \times ... \times X^n) \tag{8.1}$$

where $\times$ is Cartesian product, and $\heartsuit$ is extended as an n-ary operation. Note that each X^i is a (lazy) sequence, so this formula is very succinct. Further, we can "curry" the function $\heartsuit$ so that:

$$\heartsuit \cdot (X^1 \times X^2 \times ... \times X^n) = (((\heartsuit \cdot X^1) \cdot X^2) \cdot ...) \tag{8.2}$$

where we have made use of the associativity of $\odot$ (see eqn 6.11). The final RHS is what we will be coding. It is quite amazing to realize that the proof tree in our program is actually many propositional Bayes nets overlaid together.

8.2.2 Pure probabilistic inference, ie, Bayesian network belief propagation

Backward-chaining algorithms in binary logic are relatively familiar to us as they are explained in most AI textbooks. One of the simplest backward-chaining algorithms is the SLDNF resolution[2] used in Prolog (see §8.1.2). Originally I tried to adapt SLDNF resolution for probabilistic inference, but was doing a rather sloppy job of it. Thanks to Abram Demski who pointed out that my "Prolog trick" could actually be replaced with exact Bayesian inference via a technique known as factor graphs, our inference algorithm is now on very good theoretical foundation.

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

Factor graph algorithm

Factor graphs are a way to unify probabilistic graphical models such as Bayesian networks and Markov networks. The important thing to know is that factor graphs are isomorphic to the proof trees we construct during backward-chaining, and thus we can perform Bayesian belief propagation *in situ* in the proof tree, very conveniently. For example:

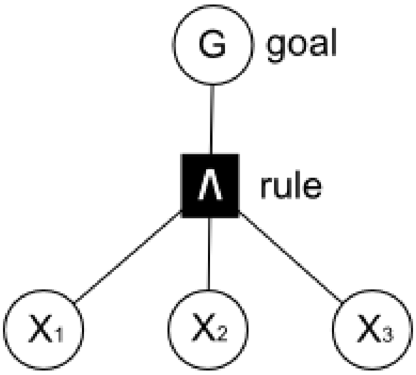


Figure 8.1: Simple proof tree / factor graph

In this example we can see that the factor node (black square) is located exactly as the "rule node" in the proof tree, therefore we might as well label the node with its operator ("$\wedge$").

The algorithm: What we need to calculate, for a "goal" variable node G, is $P(G|K)$, where K is the background knowledge. $P(G)$ is called the prior distribution of G.

When the graph has no loops, then it is a tree, and the factor graph algorithm begins from the leaves of the tree to the root which is the goal node. The leaves pass messages up to their parents, with each factor node performing a **sum-product** operation, until the messages reach the root.

[2] SLDNF stands for "Linear Selection of Definite clauses, with Negation as Failure".

Assume that we have a rule in the form:

$$H_1, H_2, \ldots \leftarrow X_1 \odot X_2 \odot \ldots \tag{8.3}$$

where the H's stand for "Heads". This is actually a compact way of stating more than 1 rule – with each head corresponding to 1 rule[3]. For simplicity's sake we assume number of H's $= 1$. For example:

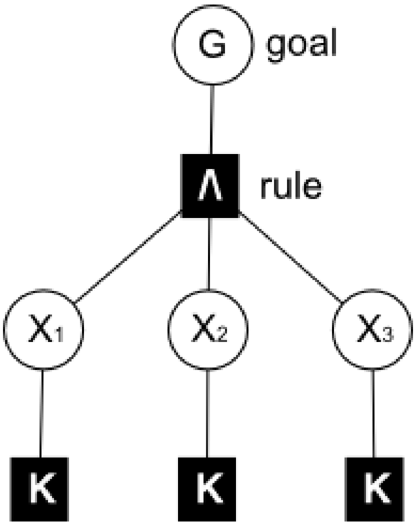


Figure 8.2: Simple factor graph in "normal" direction, with background knowledge

The link to K simply means that either the node's value is "clamped" or it is passing a message from further down the tree.

In the "normal" direction, our G is H, so the calculation is pretty straightforward. From now on we use the notational convention of omitting $P(\cdot)$ for probabilities; ie, $P(X)$ is written as (X):

$$(G|K) = \sum_{X_i} (G|X_i)(X_i|K). \tag{8.4}$$

The first factor on the RHS is the "**factor**" stored in the factor node. The second factor is the **messages** coming from each X_i node. This is a simple sum-product operation.

Next case, the direction is reversed:

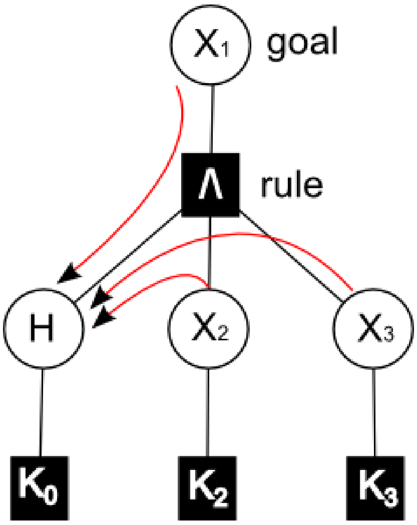


Figure 8.3: Simple factor graph in backward direction

Notice how the rule is not pointing to the goal G, and G is now taking the place of X_1. We need to apply Bayes rule twice:

[3] { To-do: explain why the multiple-head feature is desirable.}

First work out the 2-node case where $X_2, X_3, ...$ don't exist:

$$\begin{aligned}(G|K) &= \frac{(K|G)(G)}{(K)} = \frac{(G)}{(K)} \sum_H (K|H)(H|G) \\ &= (G) \sum_H \frac{(H|K)}{(H)} (H|G)\end{aligned}$$

General case:

$$\begin{aligned}(G|K) &= \frac{(K|G)(G)}{(K)} = \frac{(G)}{(K)} \sum_H (K|H)(H|G) \\ &= (G) \sum_H \frac{(H|K)}{(H)} \sum_{X_i} (H|G, X_1, X_2, ...)(X_i|G) \\ &= (G) \sum_H \frac{(H|K)}{(H)} \sum_{X_i} (H|G, X_1, X_2, ...)(X_1)(X_2)...\end{aligned}$$

Higher-order Bayesian logic

Note: here "higher-order" refers to the predicate logic aspect, not higher-order probabilities.

Essentially, what we need to do is to instantiate all possible propositional Bayesian networks with a given goal at the root, when given a set of higher-order formulas in the KB. (I'm not sure if it is enumerable, but assume that it is.)

In first-order Horn logic, the instantiation procedure seems pretty easy as it follows the SLDNF resolution algorithm (§8.1.2). With higher-order logic the task is complicated by the fact that variables can instantiate as other formulas.

Our slogan is: "One formula = one Bayesian network link = one factor graph node." Or in symbols:

formula =—▷= ■

{ To-do: Higher-order unification seems to work fine with such formulas... but I need to work out the details... }

🚧

Loopy inference

🚧

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

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

Table 8.1

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

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 ∧ ¬ 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 (§5.15.3). Then the result is negated by transforming the mean:

$$\mu' = 1 - \mu \tag{8.5}$$

Case $\wedge$: Convert the $\mathcal{P}(\mathcal{Z})$ value to $\mathcal{P}(\mathcal{B})$ via eqn (5.10). 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 $\overset{Z}{\wedge}$ $(=\min)$, $\overset{Z}{\vee}$ $(=\max)$, $\overset{Z}{\neg}$ $(= 1 - z)$ and the fuzzy modifier $\Gamma()$. At this stage we ignore Soft min and max. The $\Gamma()$'s are applied before the other operators.

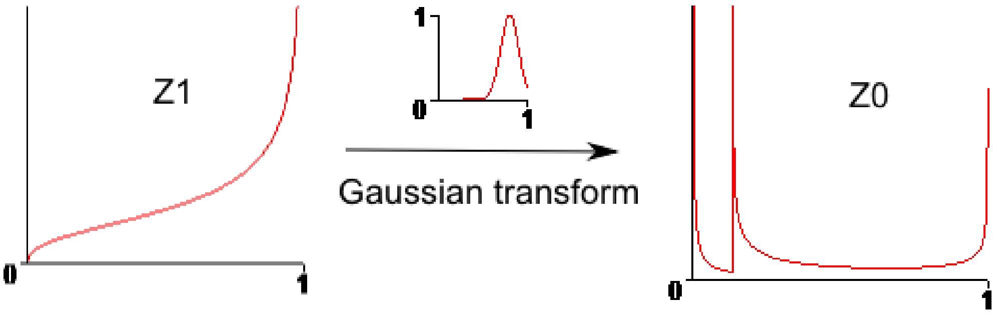
Case $\Gamma()$: If $z_0 := \Gamma(z_1)$ and z_1 is given by the probability density function $f_1(z1)$, then the probability density of z_0 would be given by:

$$f_0(z_0) = f_1(\Gamma^{-1}(z_0)) \left| \frac{d\Gamma^{-1}}{dz_0} \right| \tag{8.6}$$

which is explained in [Wikipedia, 2008]. If $\Gamma()$ is the Gaussian function given in eqn (5.7) 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):

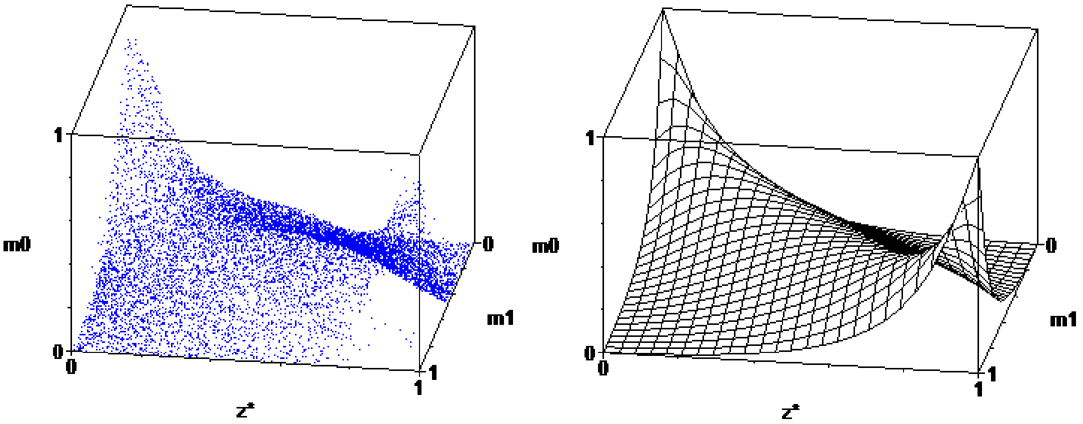


Figure 8.4: Gaussian fuzzy modifier: empirical data and result of regression

Case $\overset{Z}{\neg}$: Simply negate the $\mathcal{P}(\mathcal{Z})$ distribution by eqn (8.5) with variance unchanged.

Case $\overset{Z}{\vee}$: Given $z_0 := z_1 \overset{Z}{\vee} z_2$, 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 $\mathcal{P}$ denotes probability measure which is a set function, $F(t)$ denotes CDF's, $f(t)$ denotes PDF's.

$$\begin{aligned} F_0(t) &= \mathbf{P}(Z_0 \leq t) \\ &= \mathbf{P}(\overset{Z}{\vee}(Z_1, Z_2) \leq t) \\ &= \mathbf{P}(max(Z_1, Z_2) \leq t) \\ &= \mathbf{P}(Z_1 \leq t \wedge Z_2 \leq t) \\ &= \mathbf{P}(Z_1 \leq t)\,\mathbf{P}(Z_2 \leq t) \qquad \text{assuming } Z_1, Z_2 \text{ independent} \\ &= F_1(t)F_2(t) \end{aligned}$$

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.[4] Then we get:

$$f_0(t) = f_1(t)F_2(t) + f_2(t)F_1(t) \tag{8.7}$$

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:

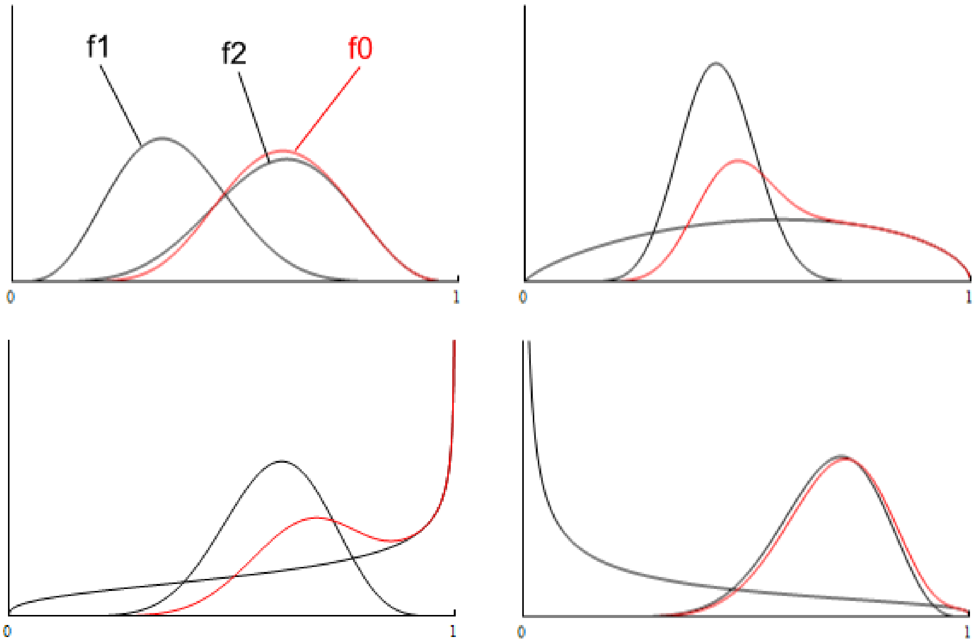


Figure 8.5: $z_0 := z_1 \overset{Z}{\vee} z_2$

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.

[4] 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:

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

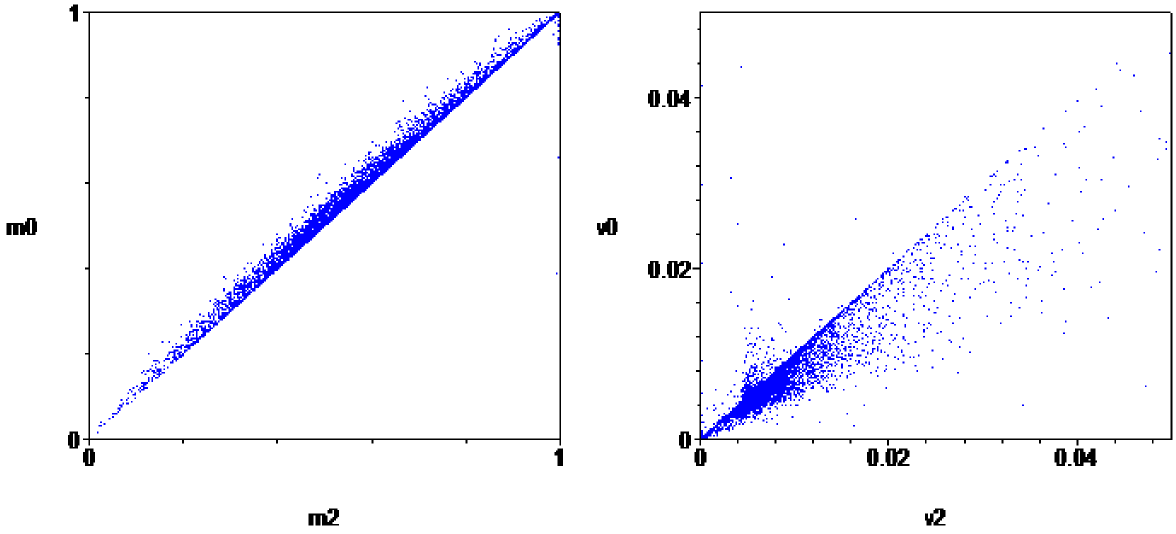


Figure 8.6: Plots of mean and variance; output vs input

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

$$f_0(t) = f_1(t) + f_2(t) - f_1(t)F_2(t) + f_2(t)F_1(t) \tag{8.8}$$

In summary, the fuzzy inference rules are:

1. For $z_0 := \neg z_1$
   $\mu_0 = 1 - \mu_1$
   $v_0 = v_1$
2. For $z_0 := z_1 \vee z_2$
   $\mu_0 = max(\mu_1, \mu_2)$
   $$v_0 = \begin{cases} v_1 & : \quad \mu_1 > \mu_2 \\ v_2 & : \quad \mu_2 > \mu_1 \end{cases}$$
3. For $z_0 := z_1 \wedge z_2$
   $\mu_0 = min(\mu_1, \mu_2)$
   $$v_0 = \begin{cases} v_1 & : \quad \mu_1 < \mu_2 \\ v_2 & : \quad \mu_2 < \mu_1 \end{cases}$$
4. 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 $\sigma_1 = k_1(m_1 + z^* - 1)^2 + k_2$
   $\sigma_2 = k_5(z^* + \sqrt{3}m_1 - k_6)^2 + k_7$
   the k's are regression parameters

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

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

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

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 (5.10). 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".

8.2.4 Inference of confidence

Confidence was introduced in §7.2. There, I explained that:

1. The confidence of rules can be measured in a frequentist way[5]
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:

$$\begin{array}{lcr} \text{A} \longleftarrow & \text{———} & \text{B} \\ ? & c_R & c_1 \end{array}$$

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)

$$\begin{array}{lcr} \text{A} \longleftarrow & \text{———} & \text{B} \\ (0) & 0 & 1 \\ \text{A} \longleftarrow & \text{———} & \text{B} \\ (0) & 1 & 0 \\ \text{A} \longleftarrow & \text{———} & \text{B} \\ (0) & 0 & 0 \\ \text{A} \longleftarrow & \text{———} & \text{B} \\ (1) & 1 & 1 \end{array}$$

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).

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

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

8.2.5 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:

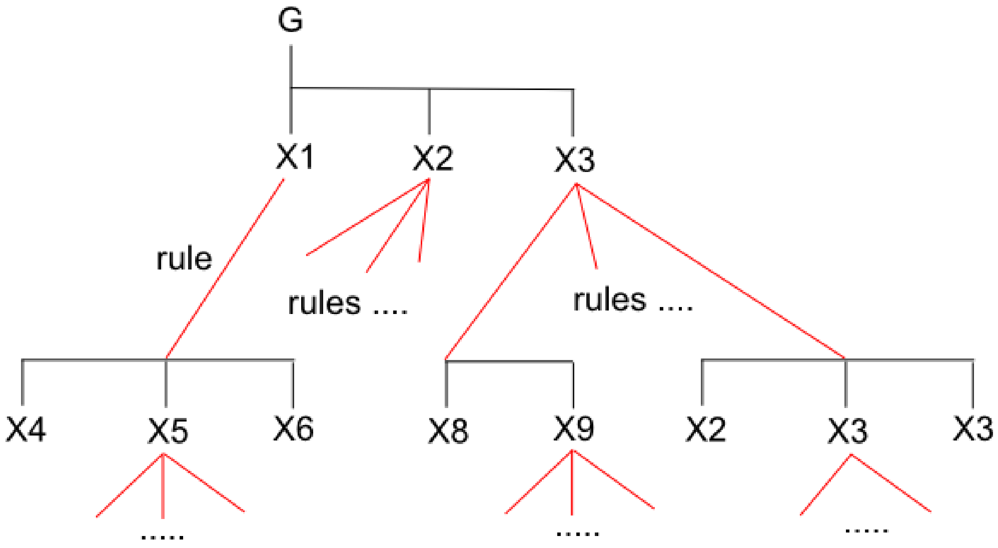
G $\leftarrow$ op(X1, X2, X3, ...)

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 $G$
repeat forever
    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 $G$)
    select an applicable rule $R$ whose head unifies with $G$
    if recursion has gone too deep or the (incomplete) proof is already too long
        return 'fail'
    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)[6]. 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.

8.3 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.

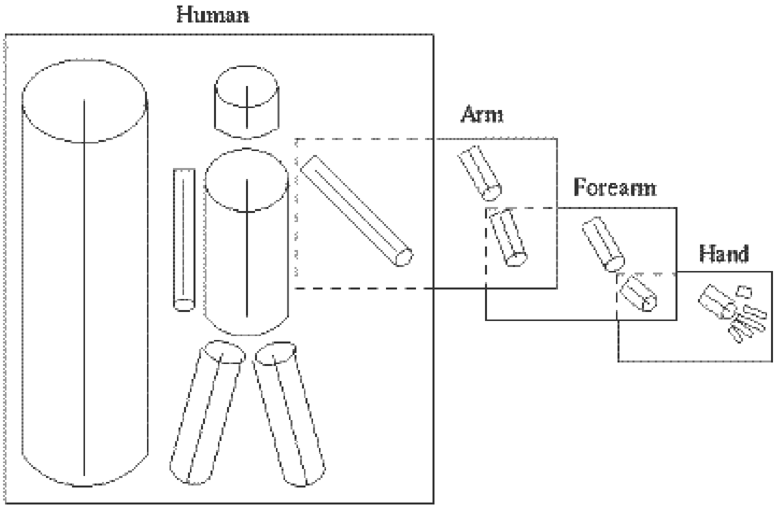
In classical logic, the algorithm for abduction is identical to deduction (backward chaining) except that the termination criterion is different. In deduction the leaves of the search tree should be ground facts in the KB (which are believed to be true). For abduction, the leaves should be so-called **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.

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

9 Pattern recognition

The diagram below is due to David Marr:



A highly simplified example of pattern recognition is the visual recognition of a human body by a Prolog rule such as[1]:

```
human :- head, torso, arm1, arm2, leg1, leg2.
```

and each body part can be further recognized by its components such as:

```
arm1 :- upper-arm, forearm, hand, fingers.
```

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.

9.1 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")

[1] Note that these concepts refer to *visual features* rather than ideas (The abstract concept of a human can be recognized via a larger set of rules that includes the visual rules). The visual features should also be related to each other by some spatial predicates, but we ignore them here. The details of visual recognition is explained in §17.

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.

9.2 Similarity

Similarity is an essential component of commonsense reasoning. For example, if we are told that "the whale is like a fish except that it is a mammal" we could draw the conclusion that the whale can swim (because fishes usually can swim) [2]. The similarity measure may also help in associative recall (§13.1).

9.2.1 From equality to similarity

Similarity ($\approx$) can be viewed as a generalization of the equality relation ($=$) with fuzzy-probabilistic truth. For example we can write

whale $\approx$ fish.

Equality ($=$) can be defined via **Leibniz's axiom of extensionality** – which states that 2 functions are identical if their extensions are the same. In higher-order logic it can be expressed as:

$$\forall x[fx = gx] \Leftrightarrow f = g \tag{9.1}$$

with types $x : \beta$ and $f, g : \beta \to \alpha$. This definition of $=$ is given by Church's type theory in 1940.

$\approx$ can be regarded as the probabilistic version of $=$. We can simply replace the $\forall$ quantification in (9.1) with the probabilistic quantification $\#$ (§6.5.2), as in:

$$\#x[fx = gx] \Leftrightarrow f \approx g. \tag{9.2}$$

9.2.2 Leibniz extensionality and intensionality

Our strategy is this: first we will formulate the most general form of semantic similarity, which we expect to be computationally intractable, then we try to approximate it.

Leibniz extensionality (9.1) defines equality for functions or predicates, but not for individual objects. For the latter purpose we have to invert extensionality to *intensionality* as follows:

$$\begin{aligned} &\text{(extensionality)} \quad \forall Z.[xZ = yZ] \quad \Leftrightarrow x = y && (9.3)\\ &\text{(intensionality)} \quad \forall Z.[Zx = Zy] \quad \Leftrightarrow x = y && (9.4) \end{aligned}$$

where I have renamed variables to stress the symmetry. The second axiom says that 2 entities are the same if every property of one is also a property of the other.

An example of intensionality (which may be more common than extensionality) is:

$$\begin{array}{l|l} \text{has-fins(whale)} & \text{has-fins(fish)} \\ \text{lives-in(whale, water)} & \text{lives-in(fish, water)} \\ \text{vertebrate(whale)} & \text{vertebrate(fish)} \\ \hline \multicolumn{2}{c}{\therefore\ \text{whale} \approx \text{fish}} \end{array}$$

[2] But we would not conclude that the whale lays eggs because mammals don't lay eggs and we know that the whale *is* a mammal, but it is only similar to a fish, and "is" is stronger than "similar to".

Notice that Z is a higher-order variable in (9.4), which can be *instantiated* as n-ary predicates with other arguments, eg:

$$Z(x) = \text{lives-in}(x, \text{water})$$

so together with extensionality this offers a very powerful way to express all possible forms of similarities, and may be regarded as a logic-based, alternative formulation of Ulf Grenander's Pattern Theory [Grenander and Miller, 2007]. But (9.3) and (9.4) require higher-order unification which is highly intractable.

In our new "combinatory" logic (§**??**), there is no need to deal with structured terms with functions, since all terms are built up solely via composition. Thus we can combine extensionality with intensionality into:

$$\text{(similarity)} \quad \forall C.\ C[x], C[y] \Leftrightarrow x \approx y \tag{9.5}$$

where C denotes a context, ie, $C[x] = C_1 x\, C_2 = c_1 c_2 ... x\, c_{i+1} c_{i+2} ...$, ie, a term with a hole in it. (9.5) is the most general form of similarity. This kind of pattern-matching is exactly the kind handled by **unification** (§**??**), that means we know, at least in theory, the inference algorithm to calculate similarities, though it would be inefficient. Now the question is to approximate it.

9.2.3 Distance metric

Our goal is to find an algorithm that calculates the distance $d(T_1, T_2)$ between 2 arbitrary logic terms. Using the sum-of-products form, first we try to define the distance $d(P_1, P_2)$ between 2 products. One way to do this may be to assign to each atomic concept c a matrix m_c, so they form a basis set in a matrix space. Then a product $P = \prod c_i$ would have a matrix value $M = \prod m_i$. Then we find the distance $d(P_1, P_2)$ by calculating $d(M_1, M_2)$ as the distance between 2 matrices, which can be defined via the inner product $\langle x, y \rangle$ [3]:

$$d(M_1, M_2) := ||M_1 - M_2|| := \sqrt{\langle M_1 - M_2, M_1 - M_2 \rangle}.$$

Next, the distance between 2 sums $\sum P_i^1$ and $\sum P_j^2$ can be defined as:

$$\sum_i \min_j d(P_i^1, P_j^2) \tag{9.6}$$

or in other words, the minimal sum of distances between pairs of products (P_i^1's and P_j^2's).

Then we can use a learning algorithm to tweak the values of the basis matrixes m_c. For example, *"cat"* is similar to *"dog"* so we can move the matrices m_{cat} and m_{dog} closer to each other. Conversely, if 2 products are known to be semantically close, we can gradually **propagate** their numerical similarity back to their constituent atomic concepts.

Question: Is it always possible to maintain the distances in such a matrix space, consistent with our notion of similarity? What would be some counter-examples?

Formulas with variables

For example: $a\, X\, b\, Y\, c$. If a variable X can take any matrix value, the product map end up in any position in the matrix space. But, we may be able to establish some (hard or soft) constraints over X, such that the resulting product would be confined in certain regions. It seems that a formula with variables would be some sort of "hyperplane" in the matrix space. We can still define the distance between a hyperplane and other matrices.

9.2.4 Examples

(A) From:

dog $\approx$ cat,

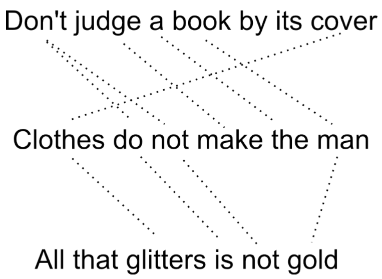
[3] The inner product between 2 matrices can be defined as $\langle A, B \rangle := tr\ A^T B$ for real matrices and $tr\ A^* B$ for complex matrices, where tr is the trace and A^* is the adjoint matrix which is equal to the conjugate transpose (alias Hermitian transpose, A^H) of A, ie, with entries $\overline{a_{ji}}$. So $d(M_1, M_2) = \sqrt{tr\ M_1^* M_2}$.

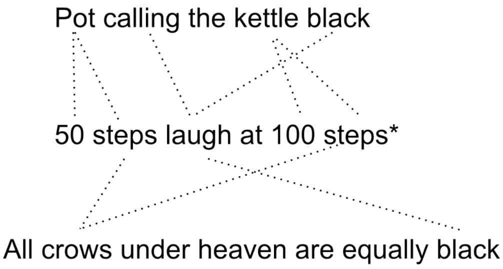
using the backward direction of (9.5) we can deduce:

let sleeping dogs lie ≈ let sleeping cats lie

but the result is incorrect and un-idiomatic. Question: Does it break our matrix-based method?

(B) The following idioms are similar in semantics but dissimilar in syntax:


Don't judge a book by its cover
Clothes do not make the man
All that glitters is not gold


Pot calling the kettle black
50 steps laugh at 100 steps*
All crows under heaven are equally black

but there are some weak / fuzzy correspondence between their words. Question: Does it break our matrix-based method? If not, how can we back-propagate the similarities to their atomic concepts?

(C) We'd be interested in measuring similarities between entities with certain structures. 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 size, mass, and temperature, etc, are.

{ The text that follows is outdated... }

What we seek is a correspondence between 2 structured entities, and to quantitatively measure the strength of such a correspondence. Perhaps we can count the number of relations common to both entities.

(D) “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:

$$\text{similarity} \quad \psi = \frac{N^{=}}{N^{=} + N^{\neq}} \tag{9.7}$$

{ 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.

NOTE: (E) is a bad example and should be replaced by something else.

(E) “Lincoln is similar to Kennedy” [4]

[*] Chinese idiom: *“Those who retreated 50 steps laugh at those who retreated 100 steps”*.

[4] 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.

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

10 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.

10.1 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.

10.2 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 $\sum P(A_i) = 1$.

10.2.1 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.

10.3 Consistency check

A few types of consistency checks are needed.

10.4 Conflict resolution

Ie, what to do when an inconsistency is found.

10.5 Theory revision

See §11.2.

11 Inductive learning

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

Note: reinforcement learning is discussed in §14.4.

11.1 "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.

11.2 What is inductive logic programming?

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]). Other notable books on induction / ILP are: [Holland, Holyoak, Nisbett, and Thagard, 1986], [Muggleton, 1992], [Lavrac and Dzeroski, 1994], [Bergadano and Gunetti, 1996], [Nienhuys-Cheng and De Wolf, 1997], [Flach and Kakas, 2000], [Lloyd, 2003], [Getoor and Taskar, 2007], [De Raedt, 2008].

Induction is the dual of abduction: both seeks a hypothesis H to explain an example e:

$H \cup KB \vdash e$
abduction seeks an H that is a ground fact
induction seeks an H that is a general rule (ie, containing variables)

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.

11.3 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) → 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: $\text{credible}(X) \wedge \text{says}(X,Y) \rightarrow Y$

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

$$\begin{array}{ll} \text{R1} \wedge \text{Y=long-hair(mary)} & \vdash \text{long-hair(mary)} \\ \text{R1} \wedge \text{Y=female(mary)} & \vdash \text{female(mary)} \\ \text{R1} \wedge \text{Y=long-hair(jane)} & \vdash \text{long-hair(jane)} \\ \text{R1} \wedge \text{Y=female(jane)} & \vdash \text{female(jane)} \end{array}$$

Thus, we can further compress the right hand sides with this rule:
 R2: female(X) → long-hair(X)
or compress what the teacher says:
 says(teacher,"female(X) → 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

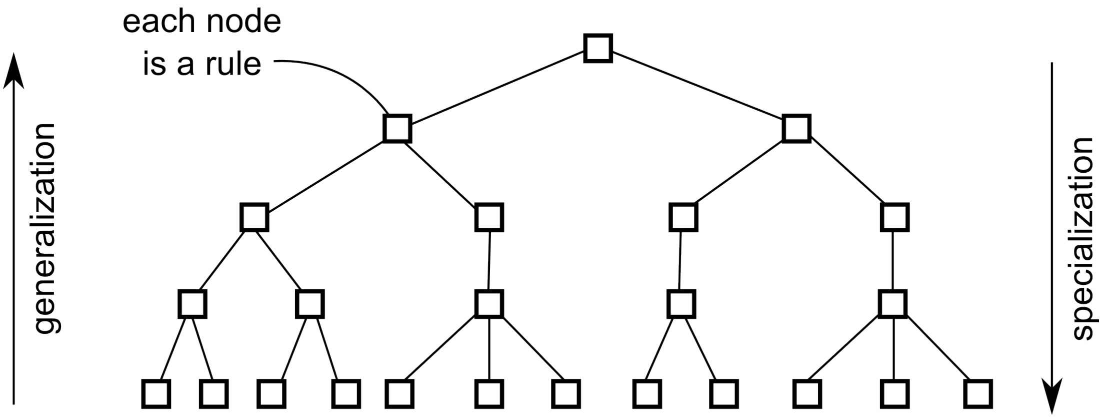
Example facts:
 Roman numeral 1 is `I`

[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.

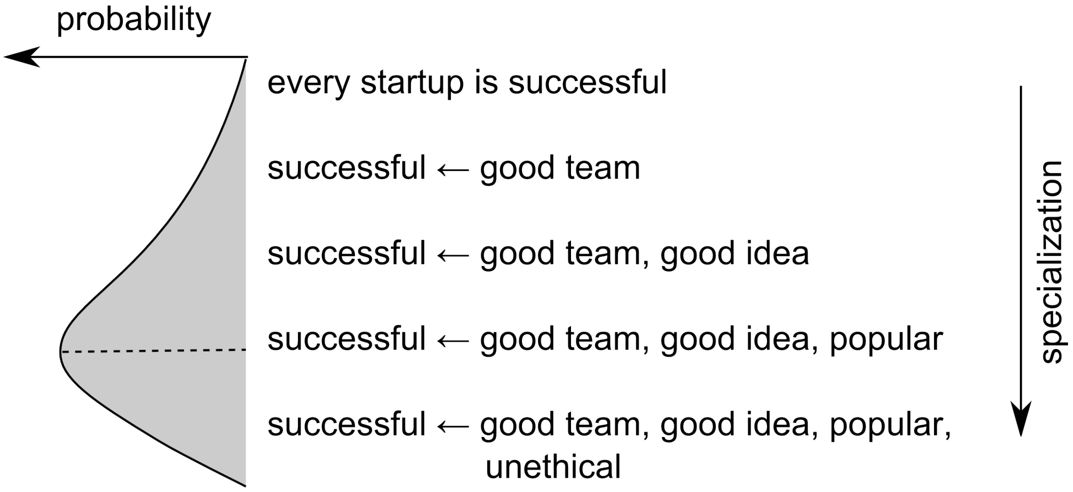
Roman numeral 2 is `II`
Roman numeral 3 is `III`
To learn:
Roman numeral n is n `I`
(which is wrong, but is reasonable in common sense).

11.4 Structure of the hypothesis space

A very important concept is that the logical hypothesis space is structured by the **subsumption order**:



An interesting phenomenon is that the probabilistic truth value of a branch of hypotheses follows a peaking pattern as we move along the specialization direction:



Actually the truth value can stay constant along a branch that specializes by adding synonymous conjuncts until such synonyms run out. Also, the truth value may actually decrease when a conjunct is added and then increase again when other conjuncts are added. For example, a non-obvious idea that spreads virally will make a startup successful, but each conjunct alone may decrease the truth value. So the search of rules cannot be pruned prematurely.

But that is the situation of rules search. During deduction our concern is of rule satisfaction. During deduction there is no need to try rules that are past the maximum, they may constitute the **frontier** of rules search.

11.5 Complexity

11.5.1 Inducing one hypothesis

The main task of ILP is to search in the hypothesis space $\mathbb{Hyp}$ specified partly by the logical syntax (such as $\mathcal{P}(\mathcal{Z})$ 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**.

11.5.2 Inducing a whole theory

Let $\mathbb{Hyp}$ denote the hypothesis space (ie, space of all logic formulae). The size of $\mathbb{Hyp}$ is huge (see above).
A logical theory T is a set of hypotheses, $\{h_i | h_i \in \mathbb{Hyp}\}$. Note that $T \in 2^{\mathbb{Hyp}}$ 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:
$T^* \vdash \{e_i\}$ and $|T^*|$ 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. }

11.6 Compression

In general, compression is achieved via **pattern recognition** (§9). Upon the arrival of new sensory input, the Pattern Recognizer is invoked to recognize patterns in the data (the process is forward-chaining). Due to the use of **fuzzy** pattern recognition, the result may be lossy, which is good.

Decompression is a special form of abduction: starting from the recognized patterns (facts) we try to abduce the original low-level sensory facts. (Though we're usually not interested in low-level facts during normal abduction.)

Inductive learning is a kind of dual of pattern recognition: whereas pattern recognition recognizes patterns using some logic rules, inductive learning learns those rules. The job of the Inductive Learner is to generalize from regularities in the sensory input whenever possible, adding new rules to the KB.

In summary:
compression = pattern recognition = forward chaining
decompression = abduction = backward chaining
learning to compress = backward chaining with rule invention

Formal definition. The goal is to find a theory T to compress a set of examples $E = \{e_i\}$ such that:

$$\begin{cases} T \vdash \hat{E} & \text{(reconstruction of memory)} \\ \hat{E} \approx E & \text{(approximation)} \\ |T| \leq |E| & \text{(smaller description length)} \end{cases} \tag{11.1}$$

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

About coverage. 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} $\rightarrow \mathbb{R}$) [2] , as a generalization of utility over states (U : {state} $\rightarrow \mathbb{R}$). 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.

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.

11.6.1 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.

11.6.2 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

[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.

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:

codelength(H) + codelength($D|H$).

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

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? }

11.7 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.

12 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)

12.1 Natural language is not essential to AGI

Our approach is to define a logic that can faithfully render arbitrary natural-language texts. We call such a logical form Geniform. The translation of NL to Geniform can itself be encoded as logical rules, thus the parsing of NL can be achieved via logical inference (as forward-chaining in a bottom-up manner). And since logical inference is reversible, the generation of NL will also come as free.

All the irregularities of NL will be taken care of via machine learning.

12.1.1 Background: 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 abstract symbolic processing similar to formal logic.

12.1.2 Background: cognitive linguistics

Our approach is closer to "formal semantics". Many authors, most famously Richard Montague, have argued that natural languages are formal languages. Some have also advocated the use of natural language as knowledge representation in AI. Thus the line between formal and natural languages is blurred.

To do: How may cognitive linguistics affect natural language processing in Genifer?

12.2 Abduction as interpretation

12.2.1 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.

Let's begin with:
"John loves Mary".

The grammar:
Sentence ::= NP VP
NP ::= Noun
VP ::= Verb NP
Verb ::= *"loves"*
Noun ::= *"John"* | *"Mary"*
Though it can only recognize 2 sentences, they are arguably the most important sentences in natural language.

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-John(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 is all syntactic 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 $\lambda x\ love(x, mary)$.

Composition of concepts. (For details see §3.3.1) Under this method, all phrases are represented by compositions. For example, the VP "loves Mary" can be represented by $app(loves, mary)$, or in short form $mary \circ loves$. This composite is a *first-order object*, ie, a first-class citizen in the logic which can be further manipulated or modified, eg, when we compose it with $john$, we get $((mary \circ loves) \circ john)$ which is equivalent to the FOL statement $loves(john, mary)$.

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.

12.3 Comprehensive grammatical categories of English

Here are some NL grammatical categories (intended to cover as broadly as possible) and typical examples in English. Designers of NL interfaces can supply translations to these examples. [The contents in this section and all the examples are taken directly from the book "English Grammar" by Peter Collins, 1998, Addison Wesley. To do: NOTE: This may be a copyright infringement but I have contacted the author without getting his reply.] This page is intended to provide a simple comparison of NL interfaces, not meant to be a standard grammar for AGI purposes.

(A) Nouns

(A.1) Common nouns

eg *"chair"*, *"chairs"*

eg *"importance"*

(A.2) Proper nouns

eg *"Australia"*

eg *"The Isle of Man"*

(A.3) Pronouns

eg *"he"*

eg *"**our** chair"* (possessive, as determiner)

eg *"theirs"*

eg *"each other"*, *"one another"* (reciprocal pronouns)

eg *"this"*, *"that"* (demonstrative pronouns)

[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)

eg *"who"*, *"whom"*, *"whatever"* (interrogative and relative pronouns)

eg *"some"*, *"both"*, *"any"*, *"each"*, *"none"*, *"nobody"* (indefinite pronouns)

(B) Verbs

(B.1) Main verbs

eg *"kick"*, *"grow"*

eg *"to kick"*, *"kicked"*, *"kicking"*, *"kicks"* (tenses and other inflections)

(B.2) Auxiliary verbs

eg *"can"*, *"have"*, *"haven't"*

eg *"She **is** dancing the lambada."*

eg *"He **has** driven for 4 hours."*

(B.3) Operators

eg *"they **will** try their best"*, *"Sam **won't** play chess"*

(C) Adjectives

(C.1) Attributive (modifies a head noun)

eg *"the **loud** bell"*

(C.2) Predicative

eg *"the bell was **loud**"*

(C.3) Gradation

eg *"**very** loud"*, *"**a bit** loud"*

(C.4) Comparison

eg *"louder"*, *"loudest"*

(D) Adverbs

eg *"The policeman acted **decisively**"*

eg *"He is a **somewhat** shy young man"*

eg *"**Actually**, many of the lifeboats have been removed"*

eg *"Patrick has lost 2 games, **however** he could still win"*

eg *"more surprisingly"*, *"most surprisingly"*

(E) Determinatives

(E.1) Articles

eg *"the"*, *"a"*, *"an"*

(E.2) Demonstratives

eg *"this"*, *"that"*, *"those"*

(E.3) Interrogatives

eg *"which"*, *"what"*

(E.4) Cardinal numerals

eg *“one”*, *“two”*

(E.5) Quantifiers

eg *“both”*, *“all”*, *“every”*, *“any”*, *“no”*, *“much”*, *“less”*

(E.6) Prepositions

(E.6.1) Time

eg *“**after** our match”*, *“**during** the exam”*

(E.6.2) Place

eg *“**in** the kitchen”*, *“**against** the wall”*

(E.6.3) Manner

eg *“**with** ease”*

(E.6.4) Agency

eg *“**by** the mechanic”*

(E.6.5) Recipience

eg *“**to** a friend”*

(E.7) Subordinates

(E.7.1) Time

eg *“**until** it rains”*, *“**as** she spoke”*

(E.7.2) Condition

eg *“**if** we win Lotto”*, *“**unless** a catastrophe occurs”*

(E.7.3) Concession

eg *“**although** she likes coffee”*, *“**while** it was undoubtedly entertaining”*

(E.7.4) Contrast

eg *“**whereas** Japan is a creditor nation”*

(E.7.5) Exception

eg *“**except** he didn’t have the strength”*

(E.7.6) Reason

eg *“**because** the bus broke down”*

(E.7.7) Comparison

eg *“you ate more peanuts **than** I did”*

(E.8) Coordinators

eg *“Peter went to Paris **but** his family stayed home”*

eg *“Barbie was wearing a new bracelet **and** a diamond ring”*

eg *“Should we come before **or** after lunch?”*

eg *“**Both** Bill **and** his wife deny the allegations”*

eg *“She **neither** hates him **nor** loves him”* eg *“It’s* ***not*** *for you* ***but*** *for me”*

(F) Phrases

(F.1) Noun phrase (NP)

eg *“large birds with sharp claws”*

eg *“writers of science fiction, who are here for a conference”*

(F.2) Verb phrase (VP)

eg *“[Lisa]* ***trains*** *3 times a week”*

eg *“There I* ***am****, minding my own business, when this tall blonde* ***walks*** *up and* ***asks*** *me for a cigarette”*

eg *“Ken said, ’I* ***have*** *enough money’ ”*

eg *“[If I]* ***could*** *have my life over again, [I]* ***would*** *not change anything”*

eg *“[They should]* ***have reached*** *the peak by now”*

eg *“[Ken]* ***is*** *building a new house”*

eg *“[Many footballers]* ***get*** *injured”*

(F.3) Adjective phrase (AdjP)

eg *“too rebellious”*, *“quite surprisingly intelligent”*, *“very slow”*

eg *“slower than a wet week”*, *“large for a goldfish”*

eg *“fiercer than I expected”*

eg *“sorry that he hurt her feelings”* (noun clause)

eg *“aware of the consequences”* (of-PP)

(F.4) Adverb phrase (AdvP)

eg *“very quickly”*, *“most reluctantly”*

eg *“more defiantly than they had predicted”*

(F.5) Prepositional phrase (PP)

eg *“****in*** *the water”*

eg *“****with*** *a spoon”*

eg *“[way]* ***below*** *standard”*, *“[just]* ***before*** *the start”*

eg *“Scott arrived in* ***a limousine*** *with* ***his girlfriend****”*

(F.6) Genitive phrase (GP)

eg *“The princess’s popularity was greater than her husband’s”*

(G) Clauses

(G.1) Object vs predicative complements

eg *“Mary contacted* ***a police officer****”* (object)

eg *“Mary was* ***a police officer****”* (predicative, *“was”*=copula)

(G.2) Subjective and objective predicatives

eg *“Indira was* ***generous****”* (subjective) eg *"We considered Indira* ***generous****"* (objective)

(G.3) Patterns of complementation

(G.3.1) intransive (S P)

eg *"Sue* ***stumbled****"*

(G.3.2) monotransitive (S P Od)

eg *"Sue* ***sipped a martini****"*

(G.3.3) Copulative (S P PCs)

eg *"Sue* ***seems drunk****"*

(G.3.4) Ditransitive (S P Oi Od)

eg *"Sue* ***offered her guests a martini****"*

(G.3.5) Complex-transitive (S P Od PCo)

eg *"Sue* ***made them drunk****"*

(G.4) Other complements

(G.4.1) PP-complements of prepositional verbs

eg *"They have decided* ***on a short vacation****"*

eg *"He approves entirely* ***of your decision****"*

eg *"Geoff protected his little brother* ***from the bullies****"*

eg *"She blames all their problems* ***on his incompetence****"*

(G.4.2) Adverbs as complement of phrasal verbs

eg *"Vera cried* ***out****"*

eg *"We give* ***up****"*

eg *"The umpires have called* ***off*** *the match"*

eg *"Please take the garbage* ***out****"*

eg *"We have come* ***up with an alternative plan****"*

(G.4.3) Non-finite complements of 'catenative' verbs

eg *"I* ***plan*** *to* ***keep*** *applying for jobs"*

eg *"You should* ***stop trying*** *to* ***get*** *invited"*

(G.5) Adjuncts

AdvP: eg *"very carefully"*, *"sometimes"*, *"moreover"*

PP: eg *"over the road"*, *"with a torch"*

Subordinate clause: eg *"because it is raining"*, *"after the music stopped"*

NP: eg *"next Easter"*, *"this Friday"*

types:

Time: eg *"at Easter"*, *"in 2 week's time"*

Frequency: eg *"each Friday"*, *"on the hour"*

Place: eg *"in Bandung"*, *"on the summit"*

Purpose: eg *"in order to test the hypothesis"*, *"for a break"*

Reason: eg *"because he is shy"*, *"on the grounds of sanity"*

Condition: eg *"if you agree"*, *"unless it is convenient"*

Manner: eg *"more politely"*, *"with a spade"*

Degree: eg *"in my opinion"*, *"to be fair"*, *"unfortunately"*

Connective: eg *"consequently"*, *"in other words"*, *"however"*

(G.6) Mood

(G.6.1) Declarative mood

eg *"He is serious"*

(G.6.2) Interrogative mood

eg *"Is he serious?"*

eg *"Anastasia has sent another e-mail, hasn't she?"*

eg *"Whose T-shirt is that?"*

eg *"Who can help me?"*

eg *"Where are my keys?"*

(G.6.3) Imperative mood

eg *"Be serious"*

eg *"Give me some more money"*

eg *"Don't forget to write"*

eg *"Let's apply for a loan"*

(G.6.4) Exclamative mood

eg *"How serious he is!"*

eg *"How many times have I asked you!"*

(G.7) Negation

clausal negation: eg *"Courtney is **not** reading"*
eg *"We have heard **nothing**"*
eg *"**Nobody** likes a loser"*
eg *"We **barely** made it"*
eg *"They **seldom** celebrate birthdays"*

sub-clausal negation: eg *"Karen is very **unfriendly**"*
eg *"The ambulance arrived **not long ago**"*

(H) Subordination and coordination

(H.1) Subordinative clause (complex sentence)

eg *"I know **that Ella lives in Sydney**"*

eg *"He knows a woman **who lives in Sydney**"*

eg *"He asked **whether anyone had submitted a nomination**"*

(H.1.1) Adverbial clauses

time: eg *"Angelica will not agree to it* ***until his attitude improves****"*

place: eg *"he goes* ***wherever his fancy takes him****"*

etc

(H.1.2) Relative clauses

eg *"Is that the boy who you were referring to?"*

eg *"I remember the days when none of us had a care in the world"*

eg *"I know who is replacing you"*

eg *"Whatever is now developing can only cause harm"*

(H.1.3) Comparative clauses

eg *"He performed worse than you did"*

eg *"Tim has won a larger grant than I have"*

eg *"She is as tall as we had anticipated"*

(H.1.4) Non-finite clauses

infinitival: eg *"Sharon wants to* ***apply next year****"*

present-participle: eg *"Sharon favors* ***applying next year****"*

past-participle: eg *"Sharon has her application* ***lodged already****"*

past-participle: eg *"Anyone* ***caught smoking here*** *can be prosecuted"*

(H.1.5) Verbless clauses

eg *"****If in doubt*** *consult your solicitor"*

eg *"He visits his parents* ***whenever possible****"*

eg *"****With Claudia in charge*** *we should be able to regain control"*

(H.2) Coordinated clause (compound sentence)

eg *"Ella lives in Sydney, but she was born in China"*

eg *"Daphne like classical music* ***but*** *her husband prefers jazz"*

(I) Information packaging

(I.1) Topic

eg *"Leonardo da Vinci painted the Mona Lisa"*
vs *"The Mona Lisa was painted by Leonardo da Vinci"*

(I.2) Focus

eg *"The judges gave near maximum points to Lipinsky"*
vs *"The judges gave Lipinsky near maximum points"*

(I.3) Weight (more weight near end of sentence)

eg *"It surprised us that an Australian skier could win a medal in the slalom event at the Nagano Olympics"*
vs *"That an Australian skier could win a medal in the slalom event at the Nagano Olympics surprised us"*

(I.4) Voice

active: eg *"Sergeant Rogerson arrested the thief"*
passive: eg *"The thief was arrested by Sergeant Rogerson"*

eg *"Each specimen was carefully dissected"*

(I.5) Cleft sentences

eg
"Whitlam recalled the remaining troops from vietnam"
"It was Whitlam who recalled the remaining troops from Vietnam"
"It was the remaining troops that Whitlam recalled from vietnam"
"It was from Vietnam that Whitlam recalled the remaining troops"

eg
"Salt air can cause rust"
"What can cause rust is salt air"
"Salt air is what can cause rust"

(I.6) Extraposition

eg *"It is a pity that CDs are so expensive"*

(I.7) Existential sentences

eg *"There is a fly in my soup"*

eg *"There's been another oil spill"*

eg *"There's a present for you"*

eg *"There followed a new round of toasts"*

(I.8) Reordering

(I.8.1) Subject-complement reversal

eg *"Tom is the short one"*
$\Longrightarrow$ *"The short one is Tom"*

(I.8.2) Topicalization

eg *"He rejects totally the corruption charge"*
$\Longrightarrow$ *"The corruption charge he rejects totally"*

eg *"It rained for most of the day last Saturday"*
$\Longrightarrow$ *"Last Saturday it rained for most of the day"*

(I.8.3) Locative inversion

eg *"Another F15 appeared from behind the clouds"*
$\Longrightarrow$ *"From behind the clouds appeared another F15"*

(I.8.4) Dislocation

eg *"Dr Davidson's never available on Wednesdays"*
$\Longrightarrow$ *"(As for) Dr Davidson, he's never available on Wednesdays"*

(I.8.5) Dative movement

eg *"Steve gave a red rose to his girlfriend"*
$\Longrightarrow$ *"Steve gave his girlfriend a red rose"*

(I.8.6) Extraposition from NP

eg *"A position for someone with programming expertise is available"*
$\Longrightarrow$ *"A position is available for someone with programming expertise"*

(I.9) Some grammatical devices

(I.9.1) Co-reference

eg *"Your sister rang yesterday. **She** asked if you could call her back."*

eg *"It is inadvisable to visit Jakarta at present. Civil unrest has broken out **there**."*

(I.9.2) Ellipsis

eg *"Come whenever you can [come]"*

eg *"Somebody should help dad. I'll ask Barry to [help dad]."*

(I.9.3) Substitution

eg *"I have 2 spare tickets for the concert. Would you like **one**?"*

eg *"She achieved less this year than she **did** last year."*

eg *"I'll collect the kids after school if you don't get the opportunity to **do so**."*

12.4 Geniform – a logical form for natural language

The Geniform logical form is special in that it can represent all natural-language grammatical categories elegantly. To illustrate this, consider:
"John loves Mary"
which is rendered in predicate logic as:
loves(john, mary)
(ignoring tense). But what if we want to say:
"Loving Mary is foolish" ?
The verb phrase *"loving Mary"* is a grammatical unit that has a specific meaning but it cannot be represented in predicate logic. In computational linguistics we usually use the λ-**abstraction** to represent it:
λx.loves(x, mary)
but as you can see, this form is not very human-readable.

Instead of λ-expressions, Geniform uses **combinators** to represent concepts, resulting in a more elegant syntax that uses only application $(a \circ b)$ and pairing (a, b) [2]. So,
"John loves Mary"
would be
((mary $\circ$ loves) $\circ$ john).
For simplicity we can omit the $\circ$ and write:
((mary loves) john).
And because of associativity we can also omit the parentheses:
mary loves john.
Notice that this *logic formula* looks exactly like English *in reverse*!

Geniform can express any NL category. For example:
chair $\Longrightarrow$ chair
wooden chair $\Longrightarrow$ wooden chair
a chair $\Longrightarrow$ a chair
chairs $\Longrightarrow$ plural chair
3 chairs $\Longrightarrow$ (3, plural) chair
3 wooden chairs $\Longrightarrow$ (3, plural, wooden) chair
any chair $\Longrightarrow$ any chair
no chair $\Longrightarrow$ no chair
Notice that (chair) does not denote *one* chair; it is just an instantiation of the abstract concept of "chair"; additional predicates are needed to specify it. For example, the application of (a) or (the) specifies that (chair) is

[2] For details about combinatory concept composition, please see §3.3.1.

singular, and the application of (3) specifies that (chair) numbers 3. It may stretch your understanding a bit to think of "any chair" and "no chair" as instances of "chair", but I think the idea is sound.

More examples:
expected $\Longrightarrow$ did expect [3]
unexpectedly $\Longrightarrow$ ly not did expect

We may (but I'm not sure yet) need a way to differentiate parts of speech, eg:
"He loves her" $\Longrightarrow$ her:pronoun loves:verb he:pronoun.

If the logic has types, we can use intrinsic types to represent parts-of-speech tags; otherwise we can use predicates to emulate types. Currently I have decided that Genifer's logic is untyped.

There may be additional problems in rendering NL into logical form:

- NL notions of *"and"* and *"or"* are not exactly the same as logical $\wedge, \vee$.
- NL notions of *"not"* are not exactly the same as logical $\neg$.
- NL notions of *"for all"* and *"there is"* are not exactly the same as logical $\forall, \exists$.

But I speculate that these minor differences can be corrected manually or via machine learning.

12.4.1 Nouns / noun phrases

Common nouns

"chair" $\Longrightarrow$ chair

"wooden chair" $\Longrightarrow$ (wooden chair)

"chairs" $\Longrightarrow$ (chair -s) $\equiv$ (s chair) $\equiv$ plural(chair)

Proper nouns

"China" $\Longrightarrow$ china

Pronouns

To do: Currently the Genifer prototype doesn't try to resolve pronoun references.

"she" $\Longrightarrow$ she

12.4.2 Verbs

Main verbs

*"John **smiles**"* $\Longrightarrow$ (present-tense smile) john

*"John **loves** Mary"* $\Longrightarrow$ ((mary, present-tense) love) john

Auxiliary verbs

*"John **has** tried"* $\Longrightarrow$ ((did try) has) john
or ((has, did) try) john
but I believe the first version is correct.

*"She **is** dancing the lambada."* $\Longrightarrow$ the lambada dancing is she

*"He **has not** slept for 2 days."* $\Longrightarrow$ ((((2, plural) day for, past-participle) sleep) , not) has he

[3] It is perhaps interesting to note that linguists believe the English past tense such as *"expected"* has evolved from the form *"expect"* + *"did"*.

12.4.3 Adjectives

Attributive

*"**young** girl"* $\Longrightarrow$ young girl

Predicative

*"Evelyn is **male**"* $\Longrightarrow$ male evelyn

It is interesting that the syntactic form for the sentence *"Evelyn is male"* and the noun phrase *"the male Evelyn"* are identical. But if we consider the sentence *"The male Evelyn won the prize"*, we will see that its logical form consists of 2 sub-statements *"Evelyn won the prize"* and *"[the] Evelyn [who won the prize] is male"*.

Gradation

*"Mary is **very shy**"* $\Longrightarrow$ very shy mary

Comparison

*"Mary is **prettier**"* $\Longrightarrow$ more pretty mary

*"Mary is **prettiest**"* $\Longrightarrow$ most pretty mary

12.4.4 Adverbs

(Tense ignored)

*"John talks **fast**"* $\Longrightarrow$ fast talks john

*"John walks **slowly**"* $\Longrightarrow$ ly slow walks john

It seems that the modifier "-ly" is redundant and the logical form can be simply:

slow walks john

12.4.5 Determinatives

Articles

*"**The** cat is black"* $\Longrightarrow$ (black, the) cat

Demonstratives

*"**This** cat is black"* $\Longrightarrow$ (black, this) cat

Interrogatives

I have not thought about the representation of **questions** in logic. This is tentative:

*"**Which** cat is black?"* $\Longrightarrow$ (black, which) cat

We need to add some sort of tag to mark this as a question rather than an ordinary statement.

Cardinal numerals

(Plurals ignored)

*"**3** mangoes"* $\Longrightarrow$ 3 mango

*"John has **3** mangoes"* $\Longrightarrow$ 3 mango has john

Quantifiers

(Plurals ignored)

*"**All** men are mortal"* $\Longrightarrow$ (mortal, all) man

*“**Most** AGI researchers are crazy”* $\Longrightarrow$ (crazy, most, agi) researcher

*“**Every** person in the room is happy”* $\Longrightarrow$ (every, the room in, happy) person

Prepositions

*“John is **in** the kitchen”* $\Longrightarrow$ the kitchen in john

*“John stands **before** Mary”* $\Longrightarrow$ mary before stands john

*“John programs **in** Lisp”* $\Longrightarrow$ lisp in programs john

*“John eats spaghetti **with** chop-sticks”* $\Longrightarrow$ (chopSticks with, spaghetti) eats john

With tense:
*“John cheated **during** the exam”* $\Longrightarrow$ (the exam during, did) cheat john

Subordinates

*“They danced **until** it rains”* $\Longrightarrow$
s_2 until s_1
+ $s_1 :=$ did dance they
+ $s_2 :=$ present rain it
where the s_i's are statements.

As a general rule, every syntactic structure must separate when they minimally qualify as a proposition, ie, having a truth value.

*“Paul will go **if** John goes”* $\Longrightarrow$
s_2 if s_1
+ $s_1 :=$ go will paul
+ $s_2 :=$ present go john

To do: There is a slight irregularity in the above translation. We could have written:
+ $s_2 :=$ future go paul.

*“Pete is taller **than** John”* $\Longrightarrow$ john than more tall pete

*“Water conducts electricity **because** it contains ions”* $\Longrightarrow$
s_2 because s_1
+ $s_1 :=$ (electricity, present) conduct water
+ $s_2 :=$ (plural ion, present) contain it

Coordinators

*“John prefers Lisp **but** Pete prefers Java”* $\Longrightarrow$
s_2 but s_1
+ $s_1 :=$ (lisp, present) prefer john
+ $s_2 :=$ (java, present) prefer pete

*“She **neither** hates him **nor** loves him”* $\Longrightarrow$
s_2 neither-nor s_1
+ $s_2 :=$ him hates she
+ $s_1 :=$ him loves she
$\Longrightarrow$ To do: ??? a bit unsatisfactory...

*“John is male **or** John is female”* $\Longrightarrow$
s_2 or s_1
+ male john
+ female john
$\Longrightarrow$ (male john) $\lor$ (female john)

12.4.6 Phrases

Noun phrase

"large birds with sharp claws" ⟹ ((sharp, plural) claw with, large, plural) bird

"writers of science fiction, who are here for a conference" ⟹
(a conference for here, science fiction of, plural) writer

Verb phrase

*"[Lisa] **trains** 3 times a week"* ⟹
((a week per, 3) times, present) train or
((a week, 3) times, present) train

*"[I] would not **change** anything"* ⟹
(anything, not) change would or
(anything, not, would) change.
The first version seems correct.

*"[Many footballers] **get** injured"* ⟹ injured get

Adjective phrase

"too rebellious" ⟹ too rebellious

"fiercer than I expected" ⟹
c_1 than more fierce
+ (c_1, did) expect i
where c_1 is an introduced constant.

"sorry that he hurt her feelings" ⟹
s_1 that sorry
+ s_1 := ((her, plural) feeling, did) hurt he

"aware of the consequences" ⟹ (the, plural) consequence of aware

Adverb phrase

"most reluctantly" ⟹ (most, ly) reluctant

"more defiantly than they had predicted" ⟹
(c_1 than more, ly) defiant
+ (c_1, past-participle) predict had they

Prepositional phrase

"with a spoon" ⟹ a spoon with

"way below standard" ⟹ (standard, way) below

Genitive phrase

"The princess's popularity" ⟹ the princess popularity
or the princess of popularity
or (the princess of, popular) nounification

To do: I'm too lazy to work out the rest of the examples...

12.4.7 Clauses

Object vs predicative complements

"Mary contacted a police officer" $\Longrightarrow$

"Mary was a police officer" $\Longrightarrow$

Subjective and objective predicatives

"Indira was generous" $\Longrightarrow$

"We considered Indira generous" $\Longrightarrow$

Complements

"Sue stumbled" $\Longrightarrow$

"Sue sipped a martini" $\Longrightarrow$

"Sue seemed drunk" $\Longrightarrow$

"Sue offered her guests a martini" $\Longrightarrow$

"She made them drunk" $\Longrightarrow$

"He approves entirely of your decision" $\Longrightarrow$

"We give up" $\Longrightarrow$

"I plan to keep applying for jobs" $\Longrightarrow$

Adjuncts

"in order to test a hypothesis" $\Longrightarrow$

Mood

"He is serious" $\Longrightarrow$

"Is he serious?" $\Longrightarrow$

"Be serious" $\Longrightarrow$

"How serious is he!" $\Longrightarrow$

Negation

"Courtney is not reading" $\Longrightarrow$

"We have heard nothing" $\Longrightarrow$

"Nobody likes a loser" $\Longrightarrow$

"They seldom celebrate birthdays" $\Longrightarrow$

"Karen is very unfriendly" $\Longrightarrow$

"The ambulance arrived not long ago" $\Longrightarrow$

12.4.8 Subordination and coordination

Subordination

"I know that Ella lives in Sydney" $\Longrightarrow$

"He knows a woman who lives in Sydney" $\Longrightarrow$

"He asked whether anyone had submitted a nomination" $\Longrightarrow$

"Angelica will not agree to it until his attitude improves" $\Longrightarrow$

"I remember the days when none of us had a care in the world" $\Longrightarrow$

"She is as tall as we had anticipated" $\Longrightarrow$

"Sharon wants to apply next year" $\Longrightarrow$

"Anyone caught smoking here can be prosecuted" $\Longrightarrow$

"If in doubt consult your solicitor" $\Longrightarrow$

Coordination

"Ella lives in Sydney, but she was born in China" $\Longrightarrow$

12.4.9 Information packaging

To do: To-do...

12.4.10 Minor word classes

Formulaic words

eg: *"okay"*, *"thanks"*, *"bye"*.

12.5 Some example English sentences

1. *"If Oswald hadn't shot Kennedy, someone else would have."* $\Longrightarrow$

s_2 implies s_1
+ s_1 := ($\underline{\text{(kennedy, did) shoot}}$, not), did) have Oswald
+ s_2 := ($\underline{\text{(kennedy, did) shoot}}$ have, did) will someone.

To do: The underlined part should be shared.
To do: The if-then syntax is not rendered naturally.

2. *"There I am, minding my own business, when this tall blonde walks up and asks me for a cigarette."* $\Longrightarrow$

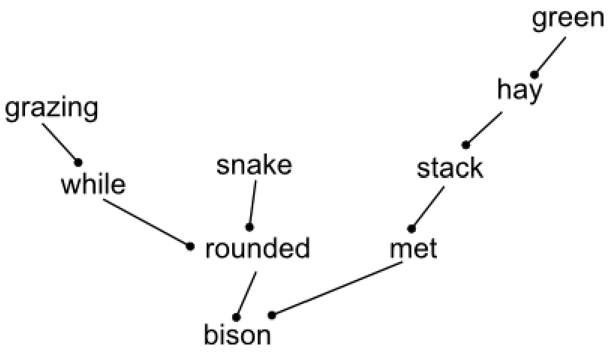
s_2 when s_1
+ s_1 := [(my own business, present) mind, there] am i
+ s_2 := [(a cigarette for me, present) ask, (up, present) walk, this, tall] blonde

3. *"Bison that rounded a snake while grazing, met a stack of green hay. But the bison waited for the hay to turn yellow before it started to eat. Once it turns yellow, hay becomes very tasty."* [4] $\Longrightarrow$

s_1 := [((gazing while, a snake), did) round, ((green hay of, a) stack, did) meet] bison
s_2 := [yellow turn to, the hay for, did] wait bison
s_3 := s_4 but s_1
s_4 := s_5 before s_2
s_5 := (eat to, did) start bison
s_6 := (yellow, present) turn hay
s_7 := s_6 once s_8
s_8 := (very tasty, present) become hay

This diagram illustrates the structure of the first formula, s_1:

[4] This sentence was made up by Ivan Vodišek during a mailing-list discussion.



Notice that the links in the graph are directed, as the application operator is non-commutative, ie,

$$a \circ b \neq b \circ a.$$

13 Memory organization and optimization

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

13.1 Introduction

We need a way to organize the KB for efficient **information retrieval**. This is critically important during inference and learning, where we need to pick some candidates for the next step of deduction or inductive learning, given the current **inference context**.

13.2 Efficient rule fetching

The current context is a set of propositions that are true currently. Together with the goal which is the query (Let us focus on deduction for the time being; the same technique that speeds up deduction will also speed up inductive learning).

An idea of Ben Goertzel's is that we can use **inference traces** to train a classifier to predict the best inference candidate. A typical training example would be given as:
1. The goal (ie, the conclusion of a proof)
2. The context or premises (ie, facts residing in Working Memory)
Try to predict:
3. The next rule involved in the proof

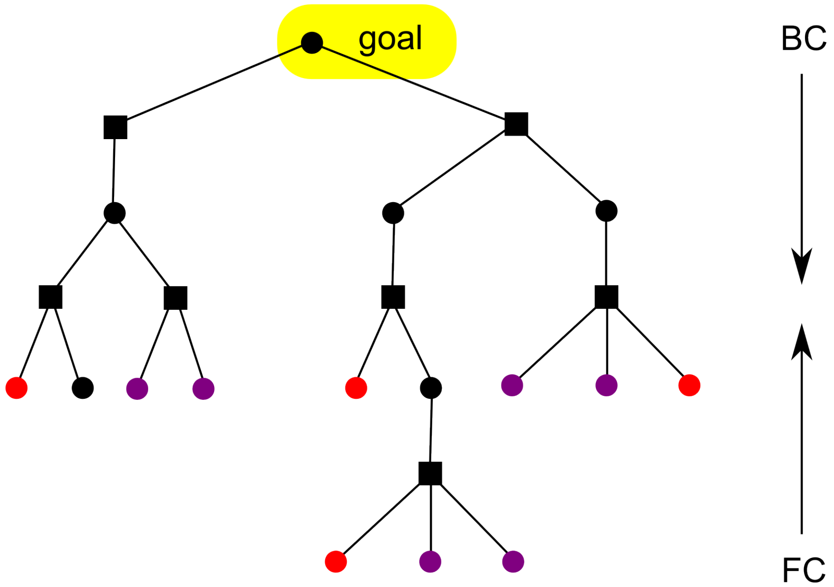
So the function we need is:

$$\{ \text{ fact } \} \times \{ \text{ set of facts } \} \mapsto \text{rule}$$

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

13.3 Objective function

Our objective is to reach the goal in a **proof tree**. All the leaf nodes must be facts drawn from KB (red nodes), with some that are highlighted by the current context (purple nodes). If we are backward chaining (BC), the goal is given. If forward chaining (FC), we can find many goals and they can be scored by **interestingness**.



The difficulty of the rule recommendation problem is that at each iteration, we do not know the ultimate score of the conclusion(s) we may reach. The situation may be similar to a chess game. Factors affecting rule recommendation are:

1. How likely is the rule to be satisfied given the current context (ie, the number of literals in the rule that remains to be satisfied)
2. Look-ahead to see what the rule can lead to, if satisfied; This is also non-static, as it depends on the current inference context
3. How much interestingness the rule can contribute to the conclusion

Or, more precisely, we want to *maximize expected interestingness*, given an inference context:

$$\mathbb{E}[\text{int}] = \sum_{Cn \in \substack{reachable \\ conclusions}} \text{int}(Cn) \cdot P(Cn | \text{ context, rule})$$

The difficulty is in estimating $P(\text{reachable conclusions}|\text{ context, rule})$. Notice that the reachable conclusions given a rule is relatively static; what varies is the inference context (and to a lesser extent the facts in KB). Indeed, given the context, we already have enough information to derive all possible conclusions.

13.3.1 Interestingness

The contribution to interestingness by a single rule can be measured by:

1. How specific / general the rule is: the most interesting rules select about half of its candidate objects; A rule that is too general or too specific is not very useful
2. How high level the rule is – a rule assigns conceptual labels to its objects, such concepts occupy certain places in the ontology

13.4 Genetic / evolutionary methods

We can construe this problem as a game:

1. The moves are the selection of rules.
2. The score is given by the interestingness of final conclusion(s) and the time needed to arrive at them.
3. Perhaps we need some way to classify inference contexts and rules, to aid the selection of rules (ie, make the game easier to play).

The genetic algorithm needs to solve these:

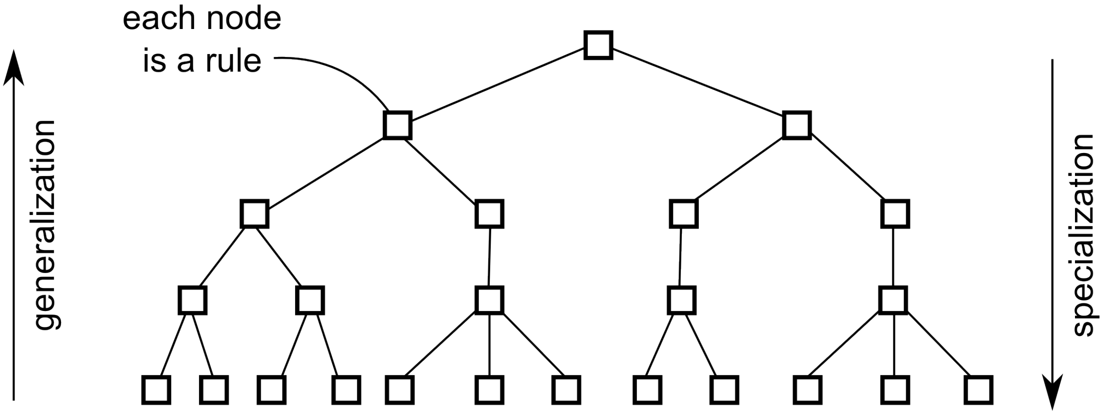
1. Organize contexts hierarchically
2. Organize KB hierarchically
3. Given context, fetch KB item, using the above hierarchies

4. Construct proof tree using unification

Alternatively 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?

13.5 KB organization

A new idea is to organize all rules in a **subsumption hierarchy**, and somehow filter the context through it. The **Rules Tree** has at its root the most general rule ("everything is true") and grows downward with successively more specialized rules.



In theory, we can embed the entire current state of the KB in the Rules Tree, by marking which conjuncts in the rules are satisfied by the current KB, but this may be impractical due to the large number of instantiations. When we have a new inference context, we can traverse down the Rules Tree to the most specialized satisfiable rules, that will yield the most specialized derivable conclusions (from that rule). Each of these conclusions will be fed through the Rules Tree again, recursively, until we reach enough interesting conclusions.

The advantage of searching the Rules Tree is that we can terminate the search on a branch early on if the parent node is past its maximum truth value (thanks to subsumption ordering and peaking, see §11.4).

This technique should be augmented with rule indexing based on current context, so it can be quickly determined which rules are possibly unifiable with the context's proposition.

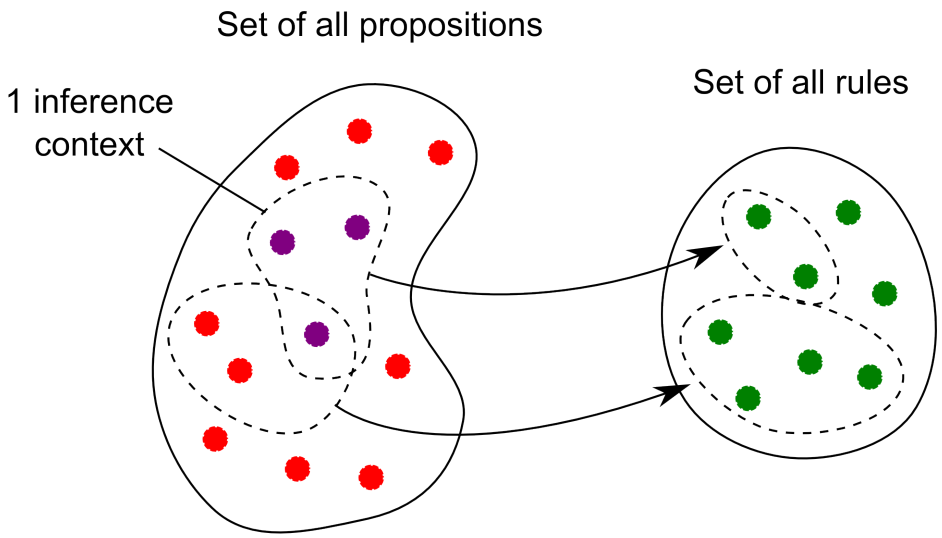
Or perhaps use a **stochastic local search** that begins in the middle of the Rules Tree.

But an interesting conclusion can be of high or low probability.

In the next iteration, we have the previous conclusion. The new conclusion depends on that. Perhaps we can back-track when stuck – but how would back-tracking increase new conclusions?

In general, is it easier to satisfy general rules first? Then we can successively refine the conclusions? But that is not true if probability is the criterion.

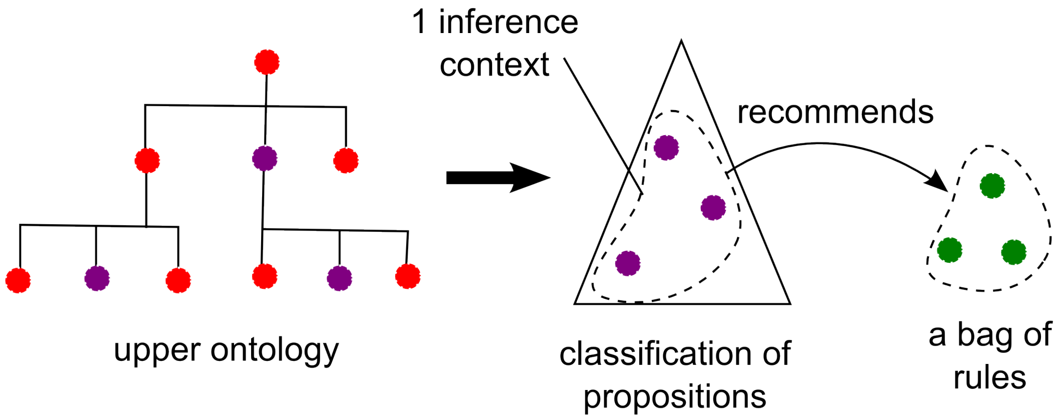
The following is an older idea. What we need is in essence a **recommendation system**: given an inference context, suggest a set of candidate rules. An inference context is a set of (complete or partial) propositions. The required mapping is **many-to-many**: multiple propositions can recommend a rule, and a rule can be recommended by more than one proposition.



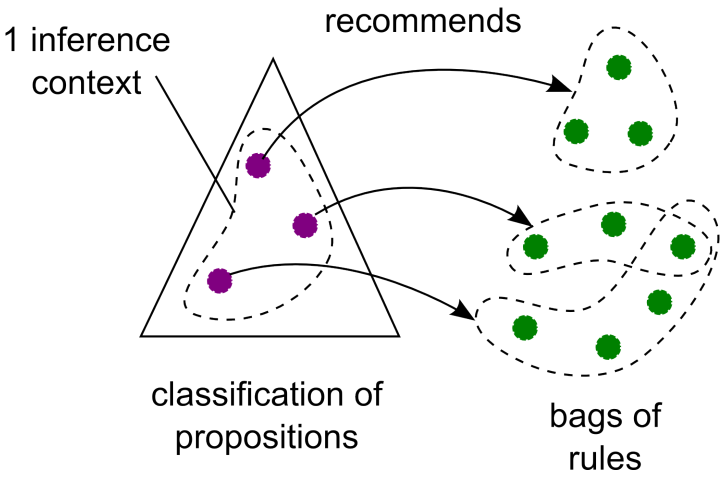
The KB is organized as an ontology (left), which we can abbreviate as a triangle (right). Each logic formula can be classified into one of the ontology nodes (red dots).

A context is usually made up of several logic formulas (either complete or partial propositions; purple dots).

The mapping we seek is:



Below is a detailed view of one possible scheme. In this scheme, *each proposition* in the context is mapped to a bag of rules. Thus we ignore the *combinatorial* effects of propositions in the context. In this scheme, obviously, for each proposition we can recommend those rules that share at least one atom with the proposition – because only those rules are unifiable with the proposition. This is the maximum set of rules that are sensible to recommend, from which we can heuristically trim down. Among these rules, only some would lead to our desired result.



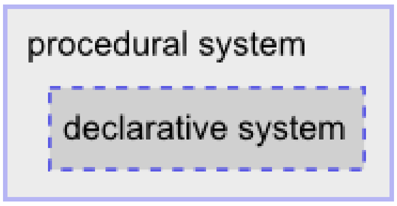
The desired choice of rules will depend on the next inference steps which are opaque to us. All we have are the clues from the current inference context (if forward-chaining), or additionally the goal (if backward-chaining).

14 Planning and acting

14.1 "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"*.

14.2 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.

14.3 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.

14.4 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:
$\pi : \mathbb{S} \to \mathbb{A}$.

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 §14.6.2 on combining deductive planning and RL.

14.4.1 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 = horny(john) ∧ ¬ good-looking(john) ∧ has-money(john)
and the policy could be a rule that recommends an action from a state:
¬ good-looking(X) ∧ has-money(X) → 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 ⊃ S1, and
   S1 → try(action1)
   S2 → try(action2)
   then both actions will be recommended when S2 is true. We need a **conflict resolution** scheme to resolve this.
   { TO-DO }

14.4.2 RL and logical reasoning

The relationship between RL and logical reasoning is very fascinating: we can regard logical deduction as the task of searching for the proof of a query; then RL can be used to perform deduction via learning the policy:
π : search state ↦ search state.
From the previous section, we see that relational RL can vastly increase the expressive power of RL through generalization of states. So the search states of the logical proof space can be generalized to what we call **cognitive states**, and the job of RL is to learn the policy:
π : cognitive state ↦ cognitive state.

In this setting, the KB is part of the environment, and we'll have actions that manipulate the KB such as adding, subtracting, and searching KB items.

Thus, the entire logical reasoner can be implemented within RL.

The question now is how to represent cognitive states and their transitions.

14.5 Means-ends analysis (MEA)

MEA is a planning method proposed first by [Newell, Shaw, and Simon, 1960] for GPS (General Problem Solver). They believed that MEA occurs in human problem solving. It is a *forward* chaining feature-space[1] search (ie, applying operators forwardly until the goal state is reached). MEA selects a difference between the current state and the goal state, then selects an operator that 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? }

14.6 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 an 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.

14.6.1 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

14.6.2 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".

[1] The feature space is similar to the state space except that a set of features only partially describe a state and thus can economically encompass many states. This is a virtue in view of an agent's limited knowledge of the complex world.

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.

15 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 §11.1, 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.

15.1 Formal program synthesis

16 Value judgments

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

16.1 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.

16.2 Sentient vs non-sentient AGI

17 Vision / perception

Note: This chapter is transferred from my older web site (2006), and needs some polishing. Many links are broken. It still represents my current view of how AGI vision should be achieved, but my focus is no longer on vision.

17.1 Design philosophy

The vision module has the following features:

- **Analytical.** Visual recognition follows a reductionistic scheme in which visual objects are broken down into smaller components (3D → 2D → 1D → 0D). There is no "black-box" in this analytical scheme.
- **Holistic.** The intelligent agent has complete access to all visual details, down to the smallest features. However this does not mean that the system is flooded with irrelevant details the attentional mechanism selects salient features in a scene, but the intelligent agent is free to focus attention on any minute detail.
- **Complete.** It has the complete range of visual functions to serve as the "eyes" of an intelligent agent.
- **Cognitive.** Visual recognition often involves reasoning. Therefore, the vision module interacts closely with the cognitive aspects of the intelligent agent.

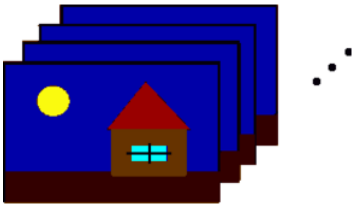
17.2 Relation between cognition and vision

1. What Is general intelligence?
2. The GI-vision approach

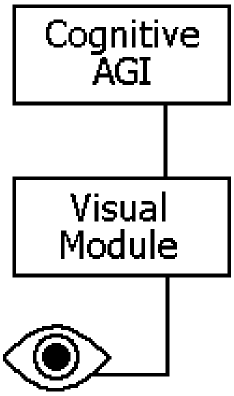
17.2.1 What is general cognition?

A fundamental idea of cognition is *compression of information*.

Sensory experience in its raw form is like a long movie:



The Visual Module compresses this movie into a semi-symbolic representation. The Cognitive Module further compresses this information.



Compression and achieved by (statistical) *pattern recognition*. For example, in the "Lena" image

the object in the center can be recognized as a "face".

GI is organized hierarchically, with higher abstraction at the top. At a very high level the Lena image may be compressed symbolically as "young woman wearing a hat".

The internal representation of a GI may be a data structure containing:

1. objects
2. properties of objects
3. relations between objects

Some examples:

1. objects: "a face", "an apple", "the letter A"
2. properties: "face.sex = female", "apple.color = red", "letter_A.font = Times New Roman"
3. relations: "hat *above* face", "apple *on* table"

AGI is a system that can automatically invent concepts of new objects, properties, and relations. These new concepts are in turn applied to the internal representation. For example it may learn the new property "rectangular" by looking at a lot of rectangular shapes.

17.2.2 The cognitive vision approach

Vision is basically the problem of compressing the sensory input stream in a meaningful way.

= "young woman wearing a hat"

But there may be several *levels* of representation.

As explained in the last section, each level of representation consists of:

1. objects
2. properties
3. relations

The pixel level is level 0 (not a representation).

Level 1 consists of basic *syntactic* features:

1. edges / lines (includes straight lines and curves)
2. regions of uniform color / intensity
3. gradients of color / intensity
4. textures

These features are represented as objects. For example:

1. object: curve1
2. property: curve1.thickness = thin
3. relation: curve1 *connected-to* curve2

At level 2 we may perform character recognition or 3D object recognition, and so on.

The key thing here is that the vision problem is solved by *combining* GI and low-level processing. Once we have represented the image as objects + properties + relations, we can assume that a GI can handle it.

Level 1 processing is hand-programmed because it starts with pixels. Then we can rely on the GI to discover higher-level features (eg the concept "cube"). The GI will also allow us to hand-program some rules (known as *knowledge insertion*), so long periods of learning may be skipped.

17.3 Machine vision — general theory and background

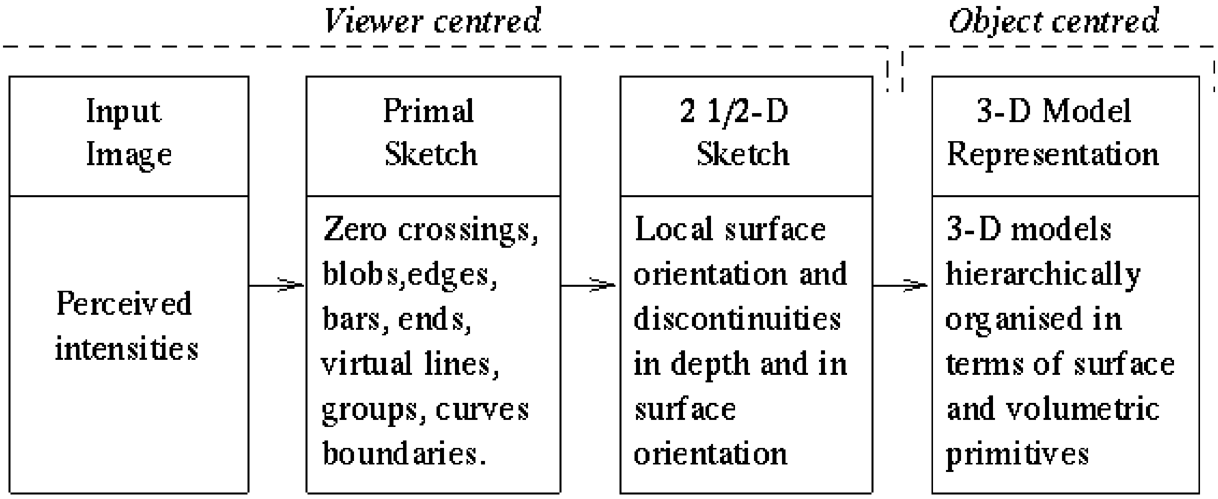
The background information on this page is adapted from [Bruce, Green & Georgeson 2003] except otherwise stated.

17.3.1 Background of general vision theories

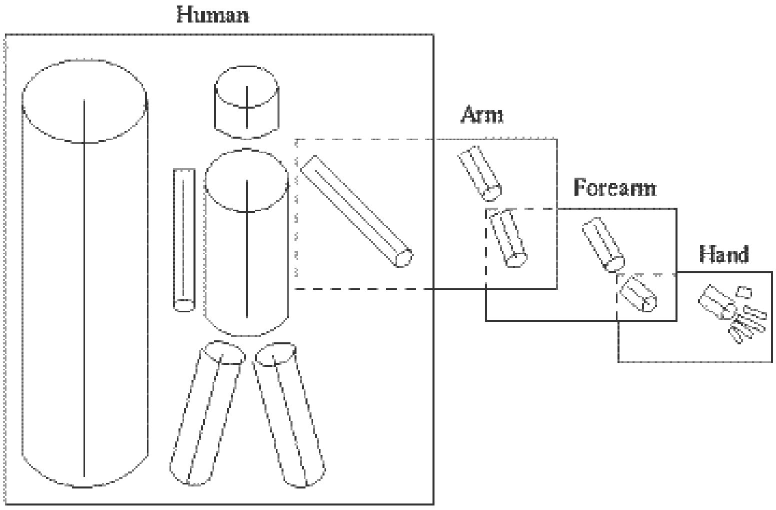
Marr & Nishihara (1978)

Many people are familiar with Marr's vision theory so I won't go into details here.

This is a diagram explaining Marr's general vision framework [from Herman Gomes' webpage]:


Viewer centred
Object centred
Input Image
Perceived intensities
Primal Sketch
Zero crossings, blobs,edges, bars, ends, virtual lines, groups, curves boundaries.
2 1/2-D Sketch
Local surface orientation and discontinuities in depth and in surface orientation
3-D Model Representation
3-D models hierarchically organised in terms of surface and volumetric primitives

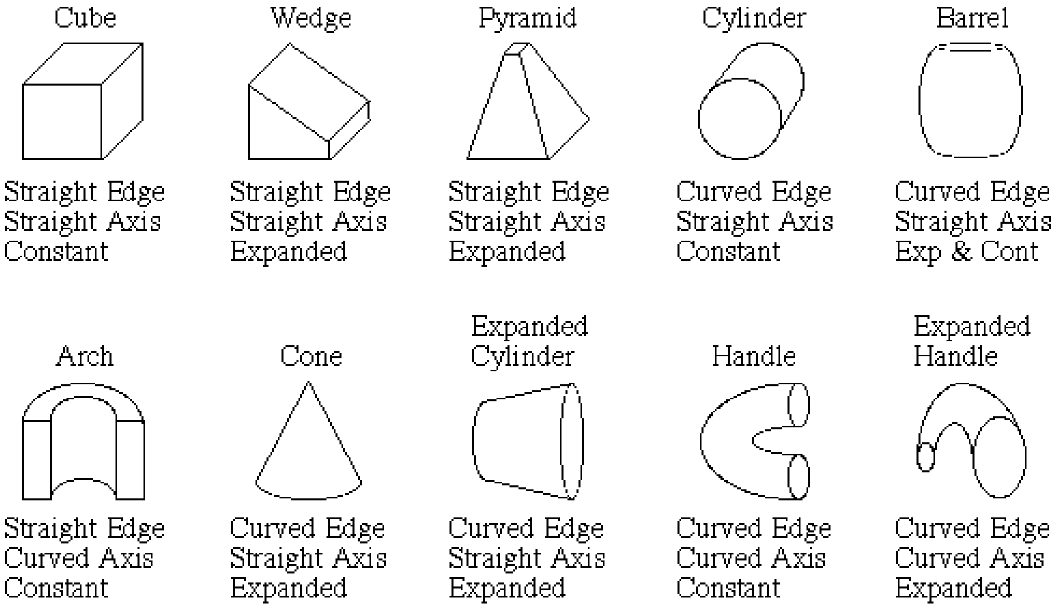
Another famous diagram from [Marr & Nishihara 1978] [redrawn by Herman Gomes]:



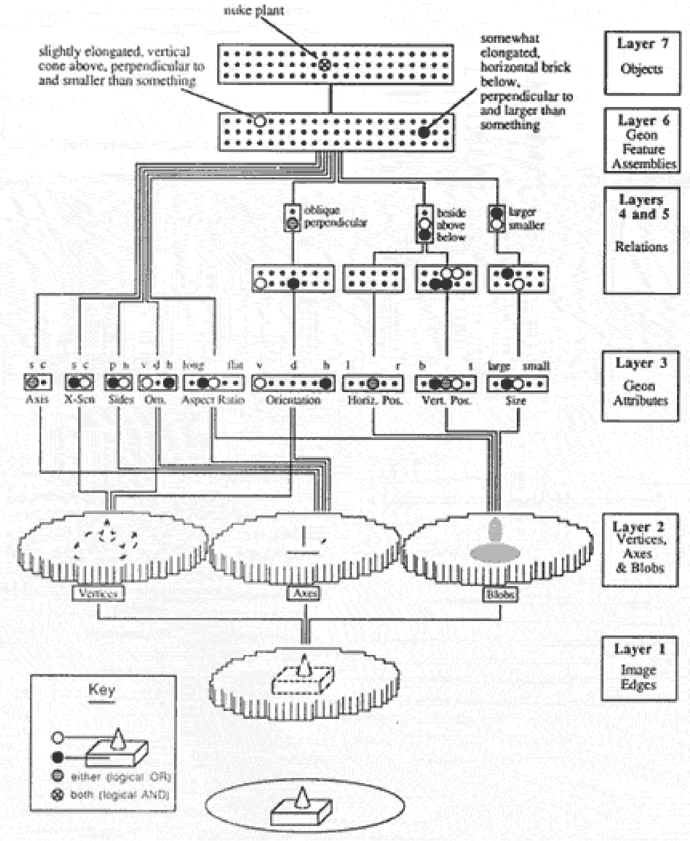
Marr & Nishihara's theory is restricted to describing objects using a set of *generalized cones* (after [Binford 1971]).

Biederman and geons

[Biederman 1987] (web page: geon.usc.edu/~biederman) proposed the "recognition by components" theory, which is closely related to Marr and Nishihara's earlier theory. In Biederman's theory, complex objects are described as spatial arrangements of basic component parts known as "geons". Geons are defined by properties that are invariant over different views. Some example geons are [taken from Kirkpatrick's web page]:



In particular, [Biederman & Hummel 1992] proposed a neural network system to recognize geon-based objects:



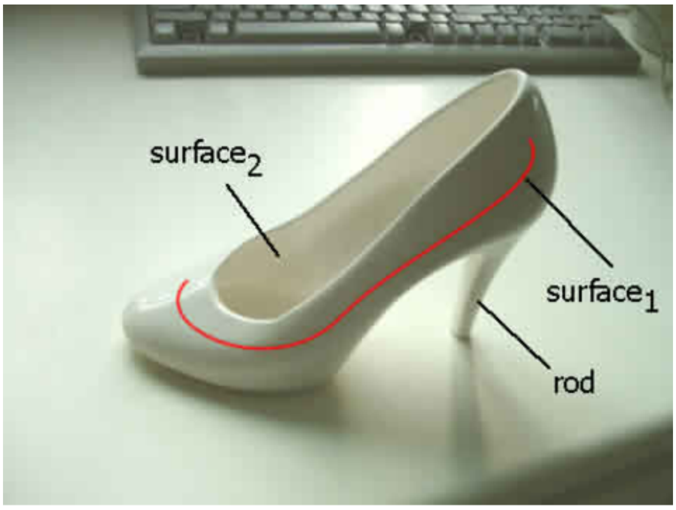
17.3.2 Our approach

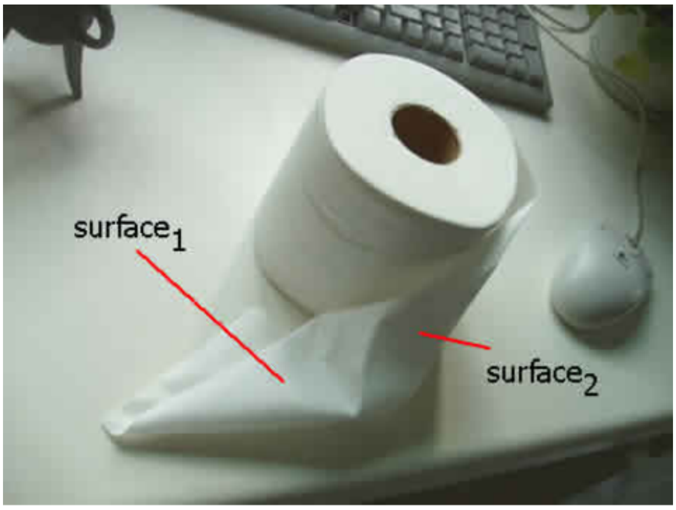
Re Marr's & Biederman's theories

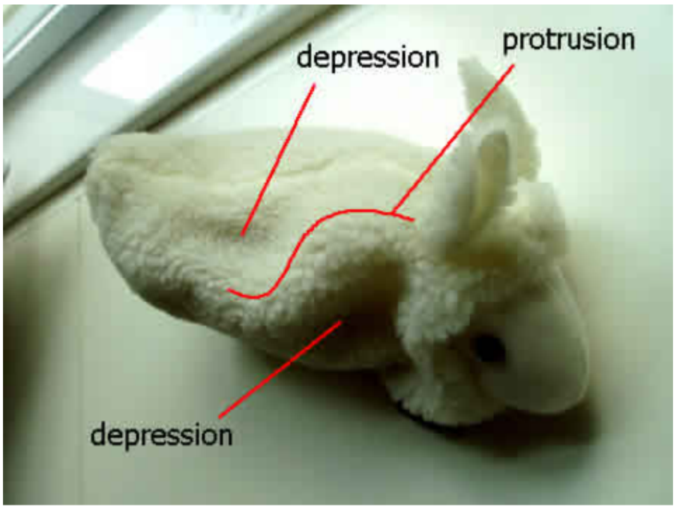
Our approach roughly conforms to Marr's framework of "primal sketch → 2.5D → 3D" reduction, but we describe the reduction as "0D → 1D → 2D → 3D". It may not be such a big difference, so I'll skip the discussion of this issue.

Another similarity with Marr is that I think the high-level representation is 3D in nature. But I do not restrict the representation to generalized cones only. My theory is that any 3D object can be defined by 2D surfaces, and this theory is not restricted to generalized cones or Biederman's geons.

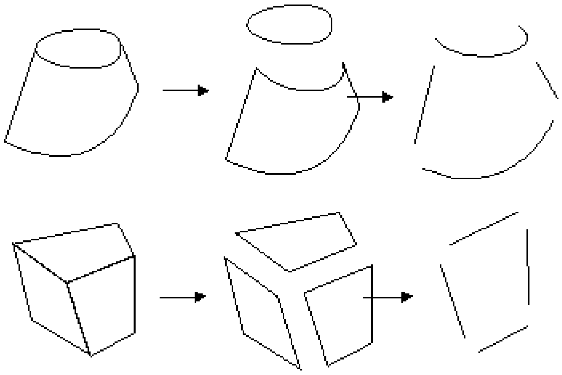
The following examples may convince you that some shapes are not representable by common geons:







In all of the above cases, the objects have parts that are defined by some irregular surfaces. The geon theory may fail in such cases because Biederman et al use neural networks to learn the geons, and the geons are *statistically* characterized by vertices, blobs, and axes. This kind of learning may be slow and recognition may be erratic. What we need is a more robust theory that can represent *any* 3D shape. The solution is to use *logical rules* to define geons in terms of 2D lines and junctions:



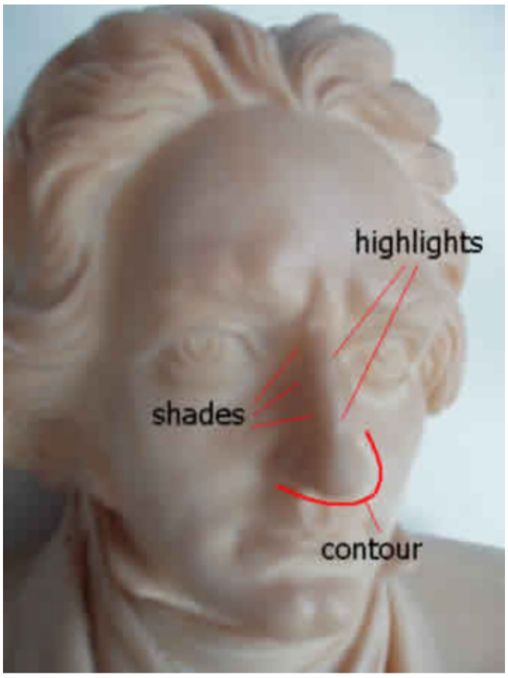
This method is more robust and can recognize things other than common geons, such as the highheel.

Shape from shading and from texture

A number of algorithms have been developed to recover "shape from shading" with the aim of describing the 3D shape given only the pattern of reflected light intensities (for example Horn & Brooks 1989), and "shape from texture". Shape from texture is a particularly hard problem so we will handle it later.

But the framework of 3-2-1-0D reduction still holds for "shape-from-X". It is relatively easy to describe 3D shapes in terms of 2D surfaces, and 2D surfaces in terms of 1D contours; but it is very difficult to jump from a set of pixels straight to a 3D description. This is probably why many prior vision theories failed.

For example, to recognize the nose, one should first recognize the shades and highlights and contours as 2D/1D features:



The conjunction of these 2D/1D elements allows us to recognize that the nose is a protrusion from the face, and its particular shape is jointly defined by the shapes of 2D/1D elements. It seems that a common mistake is to assume that the brain immediately recognizes the nose as a 3D shape from pixel-level data, without going through the intermediate stages, because the brain is usually unconscious of those intermediate stages.

Recognizing shading as 2D features will require special algorithms (so does recognizing textures). At first we will focus on using exclusively edge detection (contours) to recognize objects.

References

[Biederman 1987] *Recognition by components: A theory of human image understanding*. Psychological Review, 94, 115-145

[Biederman & Hummel 1992] *Dynamic binding in a neural network for shape recognition*. Psychological Review 99, 480-517

[Binford 1971] *Visual perception by computer*. Paper presented at the IEEE Conference on Systems and Control, December 1971, Miami

[Bruce, Green & Georgeson 2003] *Visual Perception — Physiology, Psychology and Ecology*, Psychology Press, NY

[Gomes 2000] Herman Gomes' web page: *Marr's Theory: From primal sketch to 3-D models*, http://homepages.inf.ed.ac.u

[Horn & Brooks 1989] *Shape from shading*. MIT Press, Cambridge, MA

[Kirkpatrick K] Web page on object recognition http://www.pigeon.psy.tufts.edu/avc/print/kirkpatrick/kirkpatrick_figpri

[Marr & Nishihara 1978] *Representation and recognition of the spatial organization of 3D shapes*. Proceeding of the Royal Society of London, series B, 200, 269-294

17.4 Requirements of general vision

17.5 Basic vision scheme

Our approach to solving the general vision problem is to combine vision with GI (general intelligence). See an introduction to GI-vision.

The basic tenets of my vision theory are:

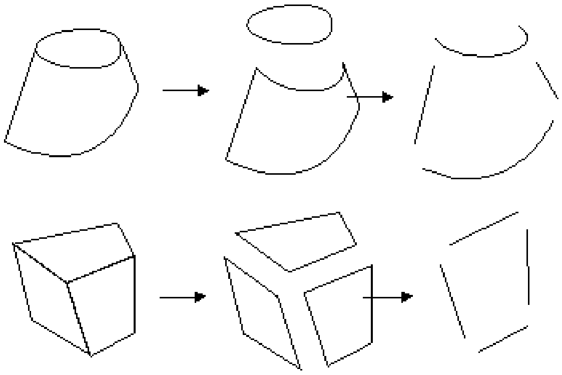
1. Break down the image via 3D-2D-1D-0D primitives
2. Apply machine learning to represent the image symbolically

I have thought about the vision problem for 1-2 years and studied many real images to conclude that "everything under the sun" can be recognized by this method. A paper will be published to explain the theory in detail.

1. 3-2-1-0D reduction
2. Logical representation
3. Machine learning
4. Example: quadrilateral
5. Approximate recognition and feedback
6. Searchlight attention
7. Relations between objects
8. Architecture of the vision module

17.5.1 3-2-1-0D reduction

The premise is that any 3D object (or its "geon-like" components) can be defined by 2D surfaces, which are in turn defined by 1D lines. Please refer to General vision theory and background for the justification of this point.



The image is first broken down into 0D/1D primitives (such as edgels and lines). Then 2D elements are recognized (such as regions and surfaces). Then 3D objects are recognized (blocks, 3D geons, etc).

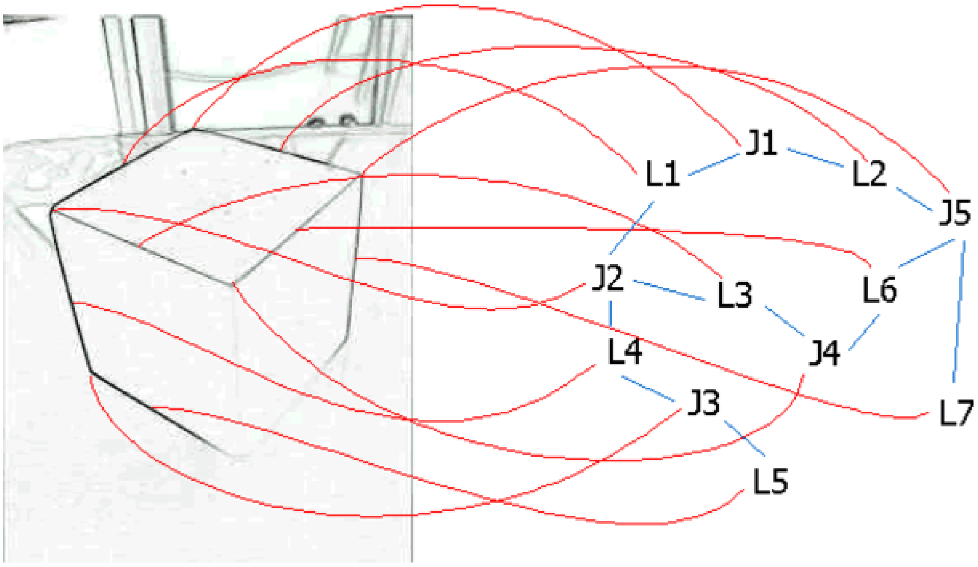
The decompositions are represented by graphical data structures (ie nodes and links). Nodes are primitive elements; Links represent the spatial relations between them.

Details of 3-2-1-0D reduction.

Our 1D-2D levels may coincide with David Marr's primal sketch level. Here we reframe these levels under the primal sketch framework.

17.5.2 Logical representation

The first step is to transform the image to a logical representation:



On the left hand side is the image; we have applied Sobel edge detection. On the right hand side is the logical representation. L1,L2,L3... = lines, J1,J2,J3... = junctions. The blue lines represent how the elements are connected. Other details at the background have not been represented.

For simple, uncluttered scenes, this scheme will work fine. The idea is that every detail, no matter how irregular, would be represented using this logical representation (using elements such as blobs and shades in addition to lines, junctions, etc).

This approach requires a lot of patience, but ultimately we would be able to analyse everything in the world. Other approaches such as neural network or SIFT are not as general or comprehensive.

We may use neural networks at the lowest level for recognizing "edgels". Then the next stage is to join the edgels to recognize longer lines.

17.5.3 Machine learning

What we need is a special kind of machine learning known as inductive learning (as opposed to deductive learning). In inductive learning a system tries to induce *general rules* from a set of observed instances ("learning by examples").

There should be an underlying knowledge representation (KR) or "calculus" that encompasses what kinds of rules are possible. The most common KR scheme is first-order predicate logic, often abbreviated as FOL (first order logic). Other options include: neural networks, semantic networks, conceptual graphs, Bayesian networks, etc.

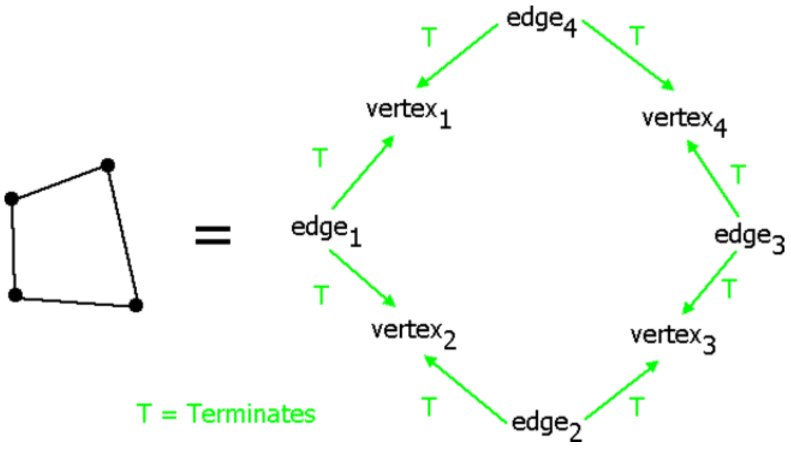
It is not easy to determine what kind of KR is adequate for our task at hand (visual recognition). Therefore we start with something simple and similar to FOL and see if it needs to be expanded or modified.

Details of inductive learning.

17.5.4 Example: quadrilateral

Any figure with 4 sides (straight lines).

Define the predicate `Terminates(edge,vertex)` to indicate when an edge terminates with a vertex.



This results in the set of logical statements:

```
Terminates(edge₁,vertex₁) = true
Terminates(edge₁,vertex₂) = true
Terminates(edge₂,vertex₂) = true
Terminates(edge₂,vertex₃) = true
Terminates(edge₃,vertex₃) = true
Terminates(edge₃,vertex₄) = true
Terminates(edge₄,vertex₄) = true
Terminates(edge₄,vertex₁) = true
```

Perhaps, we can introduce a new predicate `Connects(edge,vertex₁,vertex₂)` to simplify the above to:

```
Connects(edge₁,vertex₁,vertex₂) = true
Connects(edge₂,vertex₂,vertex₃) = true
Connects(edge₃,vertex₃,vertex₄) = true
Connects(edge₄,vertex₄,vertex₁) = true
```

Assuming that the "universe" is a connected graph that the system is currently paying attention to, now we can easily define the 0-ary predicate `Quadrilateral()` using *typed logic*:

```
Quadrilateral() :-
```

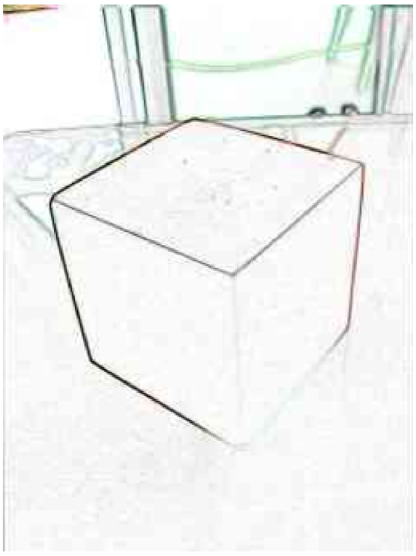
$$\begin{array}{l}
\exists\ e_1\texttt{:edge} \\
\exists\ e_2\texttt{:edge} \\
\exists\ e_3\texttt{:edge} \\
\exists\ e_4\texttt{:edge} \\
\exists\ v_1\texttt{:vertex} \\
\exists\ v_2\texttt{:vertex} \\
\exists\ v_3\texttt{:vertex} \\
\exists\ v_4\texttt{:vertex} \\
\texttt{Connects}(e_1,v_1,v_2)\ \wedge \\
\texttt{Connects}(e_2,v_2,v_3)\ \wedge \\
\texttt{Connects}(e_3,v_3,v_4)\ \wedge \\
\texttt{Connects}(e_4,v_4,v_1)
\end{array}$$

This is just an example. Please refer to this page concerning various issues of inductive learning. Some other issues specific to vision are discussed as follows.

17.5.5 Approximate recognition and feedback

One problem is that primitive features are often fuzzy and should be approximately recognized.

For example, in the image below, 2 edges and 1 vertex are almost invisible, yet given the current *context* they should be interpreted as edges and vertex. The context is important because very weak features would be regarded as noise otherwise:



The feedback mechanism should work this way: When the Recognizer finds that a concept is "almost" recognized (eg with the majority of conjuncts being true), it will select the remaining features that are not yet matched and send them to the lower-level Recognizer, which would then lower its *threshold* for recognizing those features.

This requires 2 things:

1. The Recognizer should measure a *degree of certainty* associated with each feature being recognized.
2. The Recognizer at the lower level should be able to use an feedback cue to look for certain features. This may require performing an "inversion" of the cue.

A detailed explanation of the feedback mechanism will be presented soon.

17.5.6 Searchlight attention

Another problem is that real world images are often composed of many cluttered elements, so we need to use a "searchlight" to look for individual objects in a cluttered scene.

Searchlight attention is closely related to the feedback mechanism outlined above. Due to limited computational resources we can only extract features within a "fovea" of attention. If a concept is detected to be "almost" complete, the searchlight will direct the fovea to focus on areas that are likely to complete the concept, with a concomitant decrease of attention to other areas.

This requires the searchlight to know *where* to search for the feedback cue. In a sense this also requires inversion of the cue.

17.5.7 Relations between objects

The vision system not only has to recognize individual objects, but also relations among them (represented as links).

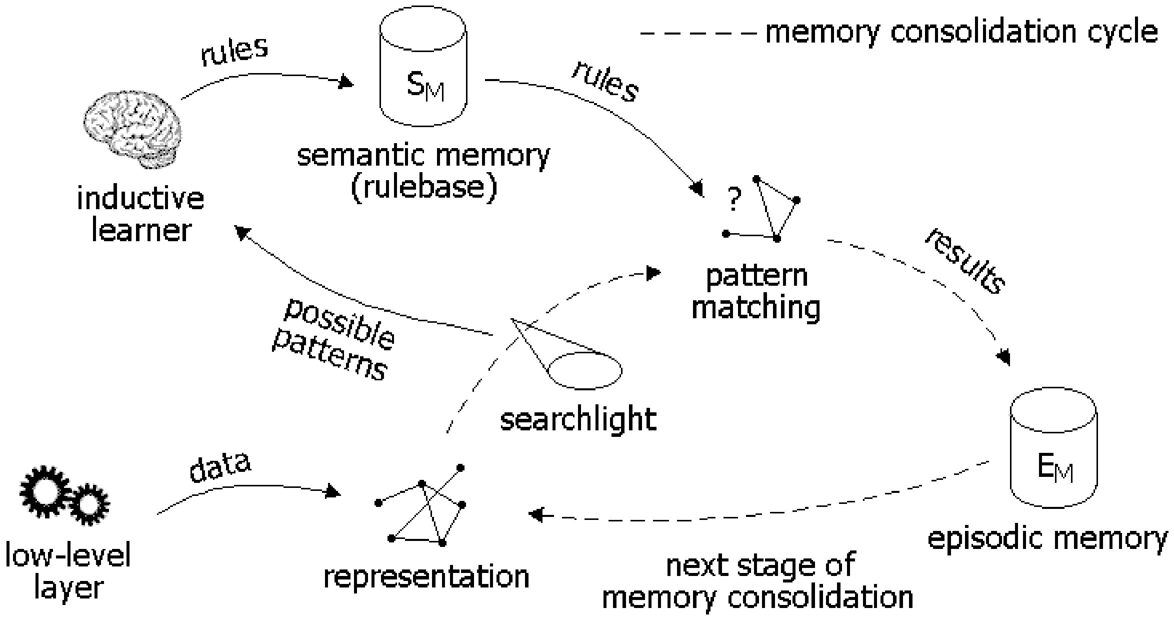
The way to achieve this is to pay attention to individual objects *sequentially*. Relations are then recognized by the identities of the objects in the sequence and by how the searchlight moved.

In our logical formulation, an object is recognized by a 0-ary predicate such as $\mathtt{Cube}_1\mathtt{()}$. Then we have to bring this object to the *next* level of recognition, where it is represented by a variable such as $\mathtt{cube}_1$. Only then we would be allowed to denote a relation like $\mathtt{Above(cube}_1\mathtt{,cube}_2\mathtt{)}$ at this level.

Recognition at each level is *independent* of recognition at other levels, except for the feedback mechanism.

The "main loop" of the Recognizer would be using the searchlight to scan around the image, and recognizing individual objects. When the searchlight moves, its movement will be recorded and later used to form the link between the current object and the next object.

17.5.8 Architecture of the vision module



The operation of the above module is typical of a rule-based system, except that there is a "loop" in the lower right corner that repeats the pattern matching process at multiple levels. What this means is that the raw sensory experience goes through multiple stages of memory consolidation via pattern matching. For details please refer to Memory systems.

This architecture has to be integrated with the larger GI (general intelligence) framework, to form a complete intelligent agent. Please refer to GI architecture.

17.6 Details of 3-2-1-0D reduction

An illustrated example ("cube") is here (incomplete)

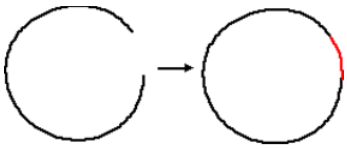
The following is a very tentative outline, the order of steps may be wrong and additional steps may be needed...

17.6.1 Stage 1: 1D Analysis

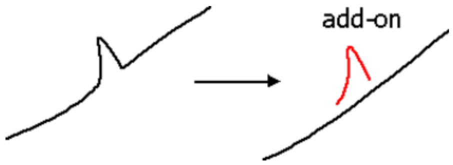
1. Edge detection — distinguish between single-pixel lines and thick lines.

2. Classification of 1D lines: straight lines, curves, junctions. From this point on the representation is a collection of discrete lines and junctions: qualitative / structural rather than quantitative / spatial / continuous.

3. Good continuation — line / curve completion.



4. Contour simplification — the image is represented by both simple and detailed contours (allowing redundency in representation).



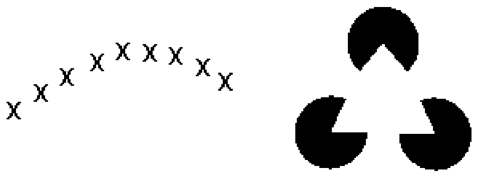
Some things can be recognized at this stage: text, handwriting, 1D graphics.

17.6.2 Stage 2: 2D Analysis

5. Characterization of regions according to color, shading or textures (segmentation).

6. Classification of 2D shapes

7. Extract gestalt-induced contours — (a) numerous small, identical / similar elements may induce contours; (b) the absence of features induces "invisible" contours



8. Deal with occlusion.

Some 2D graphics may be recognized at this stage.

17.6.3 Stage 3: 3D Analysis

9. Classification of 3D lines / surfaces.

10. Identify objects according to templates or algorithms. The question is how to define an object such as the object classes "bottle", "books", "faces", etc.

17.7 Primal sketch project

17.7.1 Background

According to Marr, the primal sketch represents changes in light intensity occuring over space in the image. It also organises these local descriptions of intensity change into a 2D representation of image regions and the boundaries between them.

Specifically, the primal sketch may include elements such as edges (curves or straight lines), color blobs, ends, junctions (of edges), texons, etc.

Marr's vision scheme consists of 3 levels:

1. primal sketch
2. 2.5D
3. 3D

My vision scheme is slightly different which goes from 3D, 2D to 1D. This difference may not be all that important; what is important is how best to recognize features at various levels, from a computational viewpoint. The primal sketch is an essential stage of any complete vision system.

What I propose is that the output of the primal sketch should be represented as a symbolic web with links that are logical predicates. Developing such a low-level layer would make a tremendous contribution towards computer vision.

17.7.2 What we're trying to do

1. Obtaining the primal sketch

One solution to the primal sketch problem (I have not surveyed the topic completely) is presented in [Gao, Zhu & Wu 2006], "Primal Sketch: Integrating Texture and Structure" (pdf), by researchers in the UCLA Center for Image and Vision Science.

As explained in the paper, an input image is separated into 2 regimes, one "sketchable" and the other "non-sketchable". The sketchable regime is one of low entropy where the image can be represented by a sparse-coding sum of primitive features; the non-sketchable regime is of relatively higher entropy where sparse-coding is no longer practical and those areas of the image should be represented as *textures*.

In our project we were trying to deal with the type of low level feature extraction in the sketchable regime, while temporarily ignoring textures. I'm currently trying to have a collaboration with the UCLA group or possibly license their technology.

2. Output an attributed graph

After obtaining the primal sketch, the second task is to represent the image as an attributed graph. This has been partly accomplished in [Han & Zhu 2005], *Bottom-up/Top-down Parsing with Attributed Graph Grammar* (pdf), for some simple shapes.

The goal of our project is to output attributed graphs similar to the above kind for all sorts of images. The expressiveness of the attributed graph may be comparable to that of first-order predicate logic.

Then the next step is to delegate the tasks of recognition and inference to an intelligent agent or cognitive architecture.

3. Handover to intelligent agent

The following tasks may be handled by the intelligent agent:

1. pattern recognition
2. attentional mechanism (eg focusing attention to various features or objects)
3. learning of concepts and facts (declarative and episodic memory)

4. complex inference (for example, a chair is "anything that can be sat on", may involve procedural memory)

The division of labor can avoid repeating these tasks for vision and for general cognition.

Currently we are considering the following intelligent agents for integration with vision:

1. Novamente (has an architecture integrating perception and cognition)
2. Soar (has provisions for sensory perception)
3. Cyc (depends on its proprietary inference engine)
4. ACT-R
5. EPIC
6. IDA (intelligent distribution agent, Stan Franklin)

Basically any general cognitive architecture that can handle sensory perception, can use our vision module.

References

[Guo, Zhu & Wu 2006] *Primal Sketch: Integrating Texture and Structure*, Computer Vision and Image Understanding, 2006 (Accepted for the Special Issue on Generative Model Based Vision) (pdf)

[Han & Zhu 2005] *Bottom-up/Top-down Image Parsing with Attribute Graph Grammar*, Statistical preprint, 2005. Submitted to PAMI (pdf)

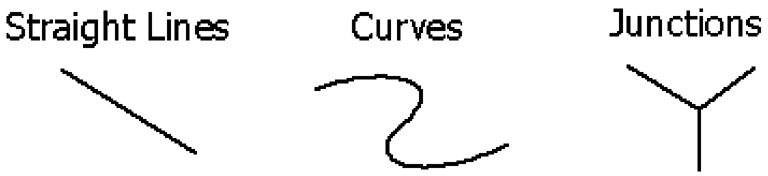
17.8 Classification of 1D Shapes

Here I attempt to classify all 1D shapes (contours) into primitive elements (lines, curves, junctions).

My view is that the vision problem (especially the recognition of 3D objects) can be solved via a 3D → 2D → 1D process. All 3D objects can be defined by 2D elements (surfaces), and all 2D shapes (areas) can be defined by 1D contours.

17.8.1 Main Classes

All 1D shapes can be decomposed into:



Properties: (thickness, color, texture)

17.8.2 1. Straight Lines

Properties: (position, length, orientation, end points)

17.8.3 2. Curves

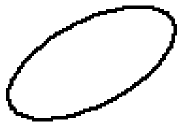
4 subtypes:

1. Simple Curves

Curves with no inflections:

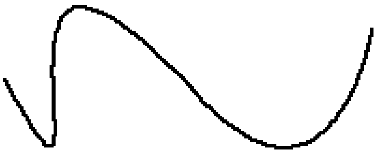
Properties: (position, length, orientation, end points, curvature)

2. Ellipses/Circles



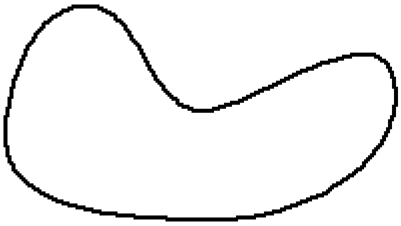
Properties: (position, orientation, size, axes ratio)

3. Complex Curves



Properties: (position, length, end points, # of inflections, curvatures)

4. Complex Loops



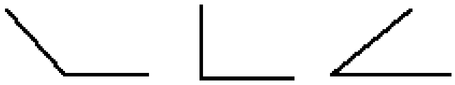
Properties: (position, area, shape description, # of inflections)

17.8.4 3. Junctions

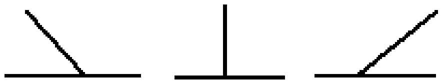
Properties: (position, # of lines, angles)

3 subtypes:

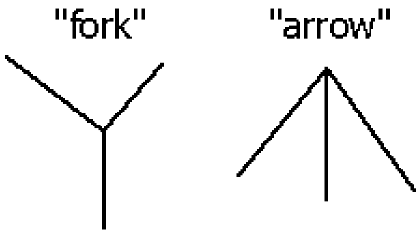
1. L



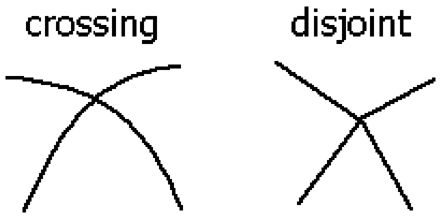
2. T



3. Y



4. X

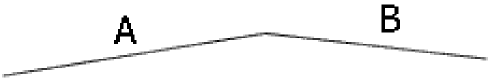


May have >4 lines of convergence.

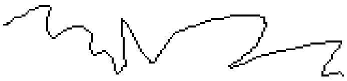
17.8.5 Examples

The following examples illustrate the concept of **self-aggregation**, which means edgel detectors (or "neurons") aggregating with others with similar attributes such as color, orientation, etc.

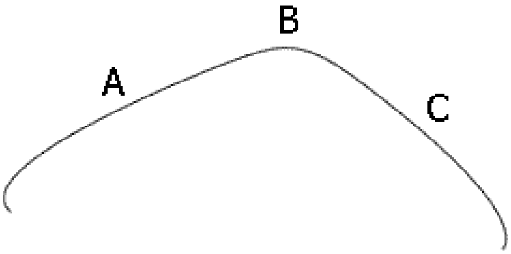
A line with a slight bend: (the bending is noticed even though it is very slight, because the A & B parts are very straight in contrast):



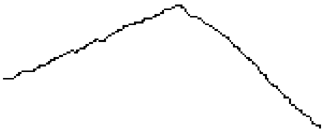
This line has many bends but is still recognized as a continuous line:



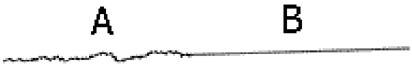
For the following line, a simple algorithm may classify it as a curve, but people will describe it as: two slightly curved lines A & C, bending at a point B. This illustrates the context-dependent aspect of human vision.



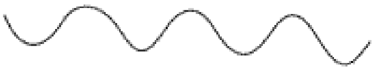
The following is yet another context-dependent example: we see 2 "shaky" but straight lines bending at an angle.



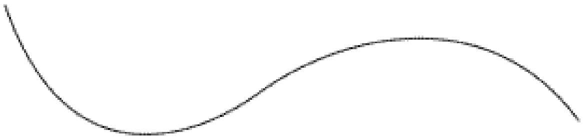
What is a "straight" line is relative: B is straighter than A but in the above context A is also considered a straight line.



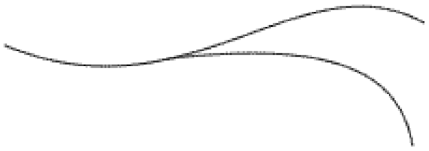
People will describe the following as a regular "sine" curve:



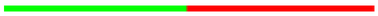
This is also a simple curve that people can describe with basic features such as crest and trough and being smooth:



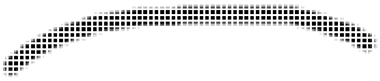
People can also characterize line junctions with special attributes, eg a "smooth" junction:



This can be considered as 1 line with 2 colors:



This is also considered a simple curve with internal texture:



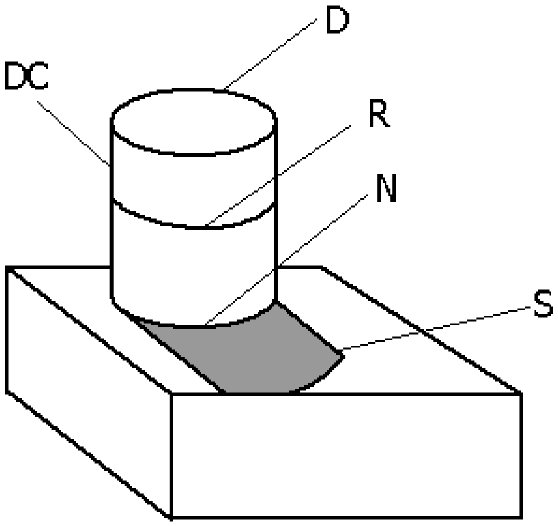
17.9 Classification of 2D features

17.10 Classification of 3D Shapes

There are a limited number of possible 3D contours and junctions. The technique of labeling all contours and junctions consistently is known as *relaxation labeling* and has grown into a large area of work.

We should work out the rules of arbitrary 3D object construction...

17.10.1 3D Edges



D = discontinuity of distance of object, discontinuity of normal of object surface

DC = discontinuity of distance of object, continuous normal of object surface

N = discontinuity of normal of object surface

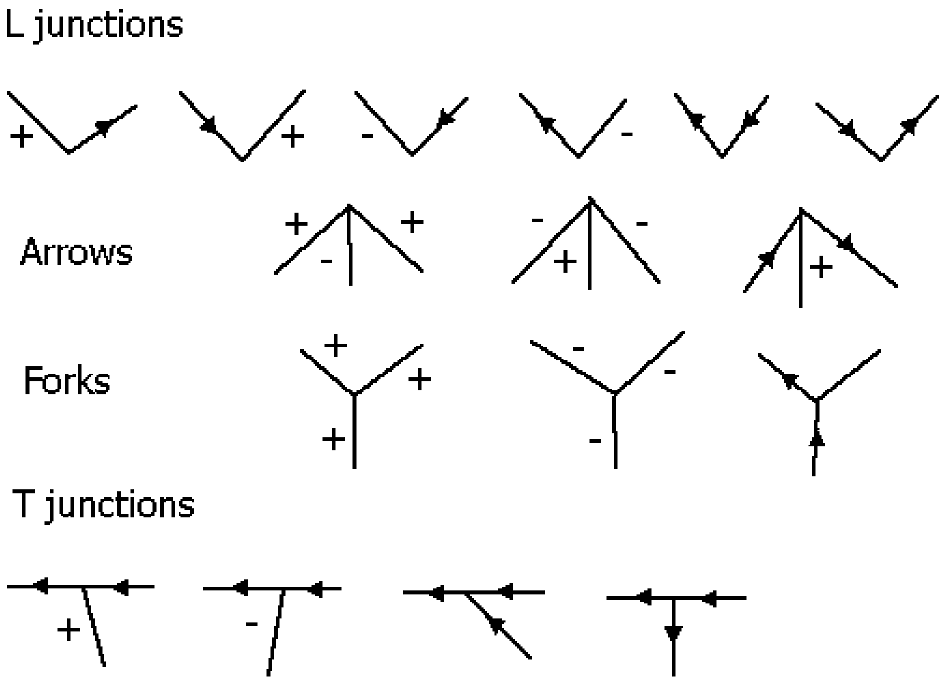
R = change of reflectance

S = shadow

Each type of contour has its characteristics such as shading. Such characteristics help to infer its type.

17.10.2 3D Junctions

There are 16 topologically possible line junctions in the "trihedral blocks world":



+ and - indicates whether a crease is convex or concave, respectively. An arrow indicates a "blade" formed by a discontinuity of distance (same as the D-type contour above). The direction of the arrow indicates which side is the *occluding* surface, which by convention is to the right.

We should extend this analysis to arbitrary 3D shapes.

17.11 Inductive learning for vision

Inductive logic programming seems to be the harder part of our project (in fact ILP is considered one of the hardest areas in computer science!)

17.11.1 Background to ILP (inductive logic programming)

Books and internet resources

1. *Machine Learning* [Mitchell 1997] has a chapter on "Learning Sets of Rules" which is a good introduction.
2. *AI: A Modern Approach* talks about ILP in chapter 19
3. This 1994 book by N. Lavrac and S. Dzeroski is free for download: *Inductive Logic Programming: Techniques and Applications*.
4. Stephen H Muggleton's website on ILP: www.doc.ic.ac.uk/˜shm/
5. Peter A Flach's website: www.cs.bris.ac.uk/˜flach and his PowerPoint on Knowledge Representation and ILP.
6. many more...

A simple ILP example

To learn the relation `Daughter(x,y)`.

A database may have the following facts:

```
Male(mary) = false
Female(mary) = true
Mother(mary, louise) = true
Father(mary, bob) = true
Daughter(bob, mary) = true
etc......
```

After presenting a large number of such facts,this rule may be learned:

```
IF Father(y,x) ∧ Female(y) THEN Daughter(x,y)
```

Existing ILP software

1. FOIL
2. CIGOL ('logic' spelt backwards)
3. GOLEM
4. PROGOL
5. LINUS
6. etc...

17.11.2 Special concerns related to vision

The "fovea" focuses on a particular area or scale of an image. Low-level feature extraction results in a graphical representation of the scene. The nodes and links of this graphical representation is then converted to a set of logical statements. This set is then taken to be the "model" (possible world) of the logic.

Low-level feature extraction has to use the searchlight mechanism to discover the spatial relations among features, and then outputs a graphical / logical representation for the next level. The "universe" of the logic at each level is distinct from each other.

17.11.3 Why use logic?

Our scheme is to reduce 3D to 2D to 1D to 0D. Along this route, many concepts need to be defined:

1. First, we have the image decomposed into edgels, which are the lowest-level features.

2. Then we define "lines" and "curves" as collections of contiguous edgels.
   (From 0D to 1D: line = conjunction of edgels)
3. Then we define "polygons", "parallelograms", "squares", etc as collections of lines.
   (From 1D to 2D: surface = conjunction of lines)
4. Then we can define "cylinders", "blocks", etc as collection of surfaces.
   (From 2D to 3D: volume = conjunction of surfaces)
5. Then we define objects such as "bicycles", "wheels", etc as composed of primitive geons.

If we can enter all these things as rules in a knowledgebase, we have a vision system that operates by the rules. Our strategy is to encode these rules as *logical formulas*.

Why not neural networks?

The use of logic to represent rules (definitions) has a tremendous advantage over neural networks. For example, a quadrilateral is defined as any figure with 4 sides. Quadrilaterals can appear in all sizes and shapes, yet we have no problem recognizing any of them. With predicate logic, we can easily define a quadrilateral in terms of edges and vertices (an example is given in Vision scheme). The resulting formula is very succinct and is capable of covering all possible cases (ie it has low algorithmic complexity).

On the other hand, in neural networks we have to learn the concept of quadrilaterals in a high dimensional space populated with many instances of quadrilaterals. Such a *statistical* learning method continually "molds" the classification space into the right shape, with incremental steps, which is extremely inefficient and prone to errors. In other words, it fails to reduce algorithmic complexity significantly.

The essense of logic is that it can represent *objects* and their *relations*. Objects are variables and relations are predicates. The use of variables allows logic to represent many things *succinctly*. For example I can use the predicate `Kicks(x,y)` to mean x kicks y. The predicate can be used to denote "boy kicks ball", "girl kicks dog", or "John kicks Mary" even though these events are very dissimilar. Thus the great expressiveness of predicate logic. Most artificial neural networks nowadays cannot express things with variables.

You may argue that the brain is a neural network and so they must be capable of achieving vision and cognition. I guess the answer is that the brain probably uses neurons to perform some sort of symbolic/logical computations, rather than statistical learning as many neuroscientists now assume. Unfortunately how the cortex performs computation is still a *terra incognita*.

17.11.4 What's the use of inductive learning?

For a very simple vision system, inductive learning is not needed. All we need is to patiently enter definitions of shapes / objects into a logical knowledgebase. Then the system can recognize those things.

Inductive learning is needed for 2 purposes:
1. teaching by "show and tell";
2. automatically discovering useful concepts.

"Show and tell"

It is far easier to show the computer a number of examples of an object (eg "pineapple") than to explicitly define the general appearance of that object. The inductive learner will automatically generalize from the exemplars.

Automatic discovery of useful concepts

This is an advanced topic, but of great practical value. A "useful" concept is one that helps meaningfully to characterize a broad class of objects. For example the concept of "leaf" may characterize the leaves often attached to many types of fruits, and may also apply to flowers and trees. So it is a useful concept.

Another useful concept is "parallelogram" which often occurs in the views of rectangular blocks. If we do not use the concept of parallelogram then the description of rectangular blocks may become cumbersome.

In other words, a useful concept reduces description lengths. But if we are talking about the total description length of a knowledgebase, then the discovery of "good" concepts requires global searching. This is therefore a computational hard problem.

17.11.5 Inductive learning methods

We are currently looking for researchers in ILP to collaborate with us to develop the Inductive Learner...

17.11.6 Outstanding questions

Do we need some special inference operations outside of predicate logic? Such as stochastic logic?

When doing higher-level recognition, does the Recognizer need to be aware of the constituents of high-level features? Eg, that a line is a collection of edgels?

17.12 Depth / distance estimation

17.13 Motion and event detection

17.14 Texture

17.15 Stereopsis

17.16 Sample images

Sample images are currently hosted at:
http://www.geocities.com/genericai/Vis-SampleImages.htm but the site will soon be closed (around October 2009). I will try to relocate them somewhere.

18 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

18.1 Choice of programming languages

18.1.1 Background: breif survey of programming languages as relates to AGI

Lisp is the single most important programming language ever invented in the history of computer science. Much of the research code in classical AI was written in Lisp, and to this day Lisp (and dialects like Scheme) remains a very practical language (Genifer's first rapid prototyping is in Lisp). Lisp was first implemented as an interpreter on an IBM 704.

Lisp is based on λ-calculus which is also important in the study of logic.

Prolog seems not as good as Lisp because it imposes the restriction of Horn clauses (over full first-order logic), and forces the programmer to use SLD resolution with a depth-first search strategy. (Though Peter Norvig pointed out this is not a severe limitation of Prolog as compared to Lisp.)

Interestingly, a logic-based AGI *itself* is like an advanced-version Prolog interpreter, enhanced with better search strategies (eg best-first search), probabilities / fuzziness (eg fuzzy Prolog), higher-order unification (as in λ-Prolog), abduction (as abductive logic programming), induction (as inductive logic programming), etc. Thus, a good understanding of Prolog is essential to the study of AGI.

ML was created by Robin Milner in the 1970s for the purpose of automated theorem proving. ML's type system helps to ensure that theorems are proved correctly. ML is used to develop the LCF (logic of computable functions) series of theorem provers, which influenced HOL, Isabelle, and HOL Light. OCaml is derived from ML. The "CAM" in OCaml stands for "categorical abstract machine" which is based on categorical combinatory logic, a variant of combinatory logic influenced by category theory.

Haskell is descended from ML, and has an elegant syntax very close to mathematics. The optimizing compiler built by Simon Peyton Jones at Glasgow makes it a very fast language in recent benchmarks. Lazy evaluation is also a strong point when implementing symbolic AI algorithms.

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. (But personally I prefer C to C++.)

C++? Not bad for AGI, in my opinion. C++ is also the choice of OpenCog.

18.1.2 Bootstrapping Genifer in Genifer

The design of Genifer is indifferent to the choice of programming languages.

My latest idea is to bootstrap Genifer in its own language.

19 Business aspects

Sure now what's worth doing, is worth doing for money.
— Wall Street (1987 movie)

19.1 Capitalism, AI, and the Singularity

Of course, it is impossible to predict things past the Singularity, but my view of the Singularity is more conservative in the sense that I believe human life will remain unchanged in some essential aspects. My premise is that capitalism will persist after the advent of AI.

19.2 My political stance

As an AGI entrepreneur, I need not be involved in politics, and I have no intention to do so, but some people are frequently trying to politicize the issue to gain an advantage over me, so I am forced to clarify a few things here.

Firstly, I am not anti-American. My goal is to create a global AGI project without bias towards a particular nationality. This does not mean that I am advocating to put an end to nationalism. The internet allows us to work across geographic boundaries, so the conditions are right for this kind of trans-national collaboration.

Secondly, I think open source as a software business model is flawed. Companies need to protect new ideas to give people incentives to innovate.

🚧

19.3 Collaborative platform

From 2010 April onwards, all accounting information of this project are disclosed.

19.3.1 Virtual credits

BitCoin.

19.3.2 Voting scheme

A. Quick start guide to AGI

This is the quickest way to get up to speed from 0 to AGI. (Warning: These recommendations are subjective!)

If you want to get a quick understanding of **neuroscience** (strictly speaking you may not even need to):
Browse, but don't read:

- a book about the human brain, such as: The Human Brain
- a textbook of neurochemistry, such as: Basic Neurochemistry
- a textbook on the neuron, such as: The Neuron
- a book on modeling a single neuron, such as: Biophysics of Computation: Information Processing in Single Neurons
- a book on modeling the whole brain, such as: Memory, Attention, and Decision-Making: A unifying computational neuroscience approach

Then you will get an idea of the complexity of wetware and a general sense of how intractable it is to re-engineer the brain (except by brute force). So we should give it up. Note the analogy: it is easier to engineer a flying machine with a novel design rather than exactly copying the bird.

🚧

To learn the basics of AI (the status quo of current AI), get one of these AI bibles:

- AIMA
- Luger
- Winston
- Nilsson

To learn logic:

- Chang & Lee
- Fitting
- higher-order logic:
- lambda calculus: ?
- combinatory logic: ?
- term rewriting: ?

Recommended books to learn programming languages (in the context of AI):

- Prolog: Ivan Bratko
- Lisp: PAIP or Winston
- ML: Paulson: ML for the Working Programmer
- Haskell: "Algorithms" by Rabhi & Lapalme

Recommended books on:

- Inductive logic programming: de Raedt
- Fuzzy logic: ?
- Bayesian networks: Pearl
- Knowledge representation: Levesque & Brachman
- Natural language understanding: Jurafsky & Martin

Symbols

$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$	classical number systems	
$\mathbb{H}\mathrm{yp}$	hypothesis space	§11.5.2
$\mathbb{P}\mathrm{rop}$	(ground) proposition space	

General logic:

$\exists, \forall$	classical existential and universal quantifiers	
$\wedge, \vee, \neg$	classical binary logic AND, OR, NOT	
$\rightarrow$	(classical) implication	§6.5
$\vdash$	entailment, syntactic	
$\models$	entailment, semantic	
$=$	equality (logic predicate)	§4.6
$\approx$	similarity = fuzzy equality (logic predicate)	§9.2
$\subseteq$	inclusion ("is-a" relation)	§**??**
$\sim$	association (logic predicate)	
$a \circ b$	composition of concepts	§3.3.1
(a, b)	pairing or union	§3.3.1
$\lambda x.Mx$	lambda abstraction	
$M : \tau$	(type theory) expression M is of type τ	
$t \overset{R}{\Rightarrow} t'$	t rewrites to t' under rewriting system R	
$t \overset{R}{\rightsquigarrow} t'$	t narrows to t' under rewriting system R	§4.5.2
$A \bowtie B$	unify(A,B)	§4.5
[s$_1$]: formula	KB stores statement s$_1$	

Fuzzy and probabilistic logic:

$\#x.Q(x)$	probabilistic quantifier ("for some")	§6.5.2
$\rightarrowtriangle$	probabilistic implication (= Bayesian network link)	§6.5
$\overset{Z}{\wedge}, \overset{Z}{\vee}$	fuzzy AND and OR	§5.8
$\overset{P}{\wedge}, \overset{P}{\vee}$	probabilistic AND and OR	§6.8
$\odot$	a (fuzzy or probabilistic) operator that combines AND and OR	§5.11
$\Gamma(\cdot)$	fuzzy modifier	§5.13
ξ	point of neutrality (fuzzy logic)	§5.7
w	total number of support for a hypothesis	§7.1
w^+, w^-	positive and negative support for a hypothesis	§7.1

Categories of truth values:

$\mathcal{B}$	binary logic
$\mathcal{P}$	(binary) probabilistic logic
$\mathcal{Z}$	pure fuzzy logic
$\mathcal{P}(\mathcal{B})$	binary-probabilistic logic
$\mathcal{P}(\mathcal{Z})$	fuzzy-probabilistic logic

Miscellaneous:

"text"	texts in English / natural language
`source code`	source code

formula	logic formulas
To do: ...	things to do

Glossary

ATMS assumption-based truth maintenance system

ATP automated theorem proving

backward chaining a deduction algorithm that makes deduction steps backwards, starting with the goal to be proved

BDI belief-desire-intention (architecture)

BN Bayesian network

CDF cumulative distribution function

CNF conjunctive normal form

DNF disjunctive normal form

DP deductive planning

EA evolutionary algorithm

FOL first-order logic

forward chaining a deduction algorithm that performs deduction steps starting from the premises

ground (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 (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

RRL relational 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 FOL, 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

Abelson, Sussman, and Sussman. *Structure and interpretation of computer programs*. MIT, 1984. URL http://mitpress.mit.edu/sicp/.

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

Andreka, Nemeti, and Sain. *Handbook of philosophical logic*, pages 133–247. Springer, 2001.

Baader and Nipkow. *Term rewriting and all that*. 1998.

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.

Bonzon, Cavalcanti, and Nossum, editors. *Formal aspects of context*. Kluwer, 2000.

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.

Brachman and Levesque. *Knowledge representation and reasoning*. Morgan Kaufmann, 2004.

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.

Craig. *Logic in algebraic form*. North-Holland, 1974.

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.

Flach and Kakas. *Abduction and Induction: Essays on their relation and integration*. Kluwer Academic Publishers, 2000.

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

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

Getoor and Taskar, editors. *Introduction to statistical relational learning*. MIT Press, 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.

Grenander and Miller. *Pattern theory: from representation to inference*. OUP, 2007.

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.

Halmos. *Algebraic logic*. Chelsea, 1962.

Halmos. *Logic as algebra*. Math Asso of America, 1988.

Hammer and Hitzler, editors. *Perspectives of neural-symbolic integration*. Springer, 2007.

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

Holland, Holyoak, Nisbett, and Thagard. *Induction*. Mit Press Cambridge, Mass, 1986.

Holldobler. *Foundations of equational logic programming*. Springer-Verlag, 1989.

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.

Lavrac and Dzeroski. *Inductive logic programming - techniques and applications*. Ellis Horwood New York (out of print but available online), 1994.

Lawvere and Rosebrugh. *Sets for mathematics*. Cambridge, 2003.

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.

McCarthy. Modality, si! modal logic, no! *Studia Logica: an international journal for symbolic logic*, 59:1:29–32, 1997. URL http://www-formal.stanford.edu/jmc/modality/modality.html.

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

Muggleton, editor. *Inductive Logic Programming. Number 38 in APIC series*. Academic Press, London, 1992.

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.

S.H. Nienhuys-Cheng and R. De Wolf. *Foundations of inductive logic programming*. Springer, 1997.

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.

Pearl. *Causality: models, reasoning, and inference*. Cambridge, 2000.

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

Plotkin. *Universal algebra, algebraic logic, and databases*. Kluwer, 1994.

Prehofer. *Solving higher-order equations – from logic to programming*. Birkhauser, 1998.

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.

Shafer. *The art of causal conjecture*. MIT, 1996.

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

Snyder. *A proof theory for general unification*. Birkhauser, 1991.

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.

Streicher. *Domain-theoretical foundations of functional programming*. World Scientific, 2006.

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.

Williamson. *Bayesian nets and causality*. Oxford, 2005.

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.

Index

Acknowledgements

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