CCPS 721 Artificial Intelligence Extras

This document contains additional notes and observations to supplement the main reading material in the course CCPS 721 Artificial Intelligence I, as taught by Ilkka Kokkarinen for the Chang School of Continuing Education, Ryerson University, Toronto.

Imperative languages vs. Prolog

"*A language that doesn't affect the way you think about programming, is not worth knowing.*" (Alan Perlis, epigrams)

"*Prolog is so simple that one has the sense that sooner or later someone had to discover it. Why did we discover it rather than anyone else?*" (Alain Colmerauer, source)

"*I used Prolog in a comparative languages course. The biggest program we did was a map-coloring one (color a map with only four colors so that no bordering items have the same color, given a mapping of things that border each other). I say biggest because we were given the most time with it. I started out like most people in my class trying to hack the language into letting me code a stinking algorithm to color a stinking map. Then I wrote a test function to check if the map was colored and, in a flash of prolog, realized that that was really all I needed to code.*" (source)

Familiar imperative languages	**Prolog**
Program statements are executed in order only going forward, never going backwards. Each statement either succeeds or crashes the program (possibly with an exception that can be caught to continue execution).	A query can succeed in zero or more ways, and **generates** these solutions one at a time, each time binding the variables that appear in it to that particular solution. When no more solutions remain, the execution **backtracks** to the previous **choice point** that still has more solutions remaining in the search tree.
Variables (usually **typed** at compile time) always refer to exactly one value of that type at any given time that they exist. They remember no history about where that value came from. Previously assigned variables can be reassigned to new values at any time, which causes them to forget their old values.	**Untyped** variables can be in two states, **unbound** or **bound** to some arbitrary **expressions** (called **terms** in Prolog lingo) Individual literals such as 42 are merely special cases of expressions for which the expression tree consists of a single root node. This expression can also contain unbound variables that can later be bound to some other expression. A bound variable cannot be bound to something else until it gets **unbound** by the Prolog execution backtracking.
All expressions are evaluated **eagerly**, with subexpressions always fully evaluated before the	Expressions that variables are bound to are kept as full expression trees, and are evaluated only when

outer expression is evaluated using the results of these subexpression evaluations.	explicitly instructed to do so with the Prolog predicate named `is`.
Assignment eagerly evaluates the right hand side and assigns the result to the left hand side variable, erasing the previous value of the left hand side variable.	**Unification** between any two expressions finds the most general way to assign the unbound variables so that the resulting expressions are syntactically equivalent. Unification is neither assignment nor an equation solver, and is performed on the structure of the **expression tree**, not on the string output form of the expression.
Program and method execution cannot be reversed from results to arguments, but can only ever proceed forward. You have to write a whole separate method to perform the operation in reverse.	Logical queries make no distinction between "argument" and "result" parameters, so each query could in theory be fully reversed. However, for efficiency reasons integer multiplication and similar operations are performed using irreversible low level operations, and the predicates using them require some arguments to be bound at the time of the query.
Code and data are fundamentally different things. Code cannot treat code as data or data as code. Code is set in stone at compilation, and in statically compiled languages, no new code can be created dynamically during program execution. Even in dynamic languages, pieces of code cannot be assigned to variables or passed to functions.	Prolog is **homoiconic**, so that rules and predicates are also expressions that other Prolog rules and predicates can handle as expressions that they receive as arguments. This makes writing **higher order predicates** and their rules not only possible but quite easy.
All computations are **irreversible** so that each function can be executed only forward from arguments to results, but not backwards from results to arguments.	Since the result of a "function" implemented as Prolog rules is merely one parameter in the relation, computations are **reversible** from results to arguments, assuming that these **relational** predicates do not internally use irreversible steps such as `is/2` for efficiency.

Equality predicates in Prolog

Operator	Negation	Explanation
`=`	`\=`	Unify LHS and RHS (with most general unifier) without the occurs check in O(n) time. Use the predicate `unify_with_occurs_check/2` for the full O(n^2) time sound unification. Most of the time, the appearance of `=` inside your rule is redundant and should be done in matching the query to the LHS of the rule.
`is`		Evaluate RHS, then unify with LHS.
`=:=`	`=\=`	Evaluate both LHS and RHS, then compare arithmetic equality.
`=@=`	`\=@=`	Term structural equality by renaming variables but without unification.
`==`	`\==`	Term equality without performing renaming or unification first.
`=..`		Break LHS into a list of functor and arg expressions, then unify with RHS. Useful in functions that operate on and generate arbitrary Prolog expressions.
`#=`	`dif`	After `use_module(library(clpfd))`, these operators can be applied to variables from integer domains to add the necessary and sufficient constraints under which the equality becomes true. For example, the query `X * X #= 16.` succeeds with the resulting constraint `X in -4\/4`. Subsumes both existing operators `=` and `is`, and internally reduces to those built-ins whenever appropriate, so when dealing with integer domains, you could always use this operator, and the corresponding comparisons `#<` etc.

Rational agents

Of the four possible combinations of human/rational with thought/action, this course examines only the combination "rational action", since after all, only externally observable actions really matter in anything. Artificial intelligence now boils down to **mechanized decision theory**. A **rational agent**, either artificial or natural, **acts** embedded in some **environment** (typically some **microworld** nested inside the complete reality) to maximize some **performance measure**.

Some salient points about what makes a problem to be part of the realm of artificial intelligence, as opposed to just ordinary engineering of non-intelligent machines:

- There exists some part of the environment that is not part of the agent so that it would fall directly within the agent's control, so that the agent cannot win by simply assigning everything to be whatever it wants.

- The agent receives some complete or partial **observations** about the current state of its environment, and has to choose the action based on these observations.
- The agent generally has a **choice** of **at least two mutually exclusive actions** to choose from. If there is no such choice of actions, the problem is merely engineering, not artificial intelligence, regardless of the intricacy of that action and the mechanics involved.
- Actions have direct or indirect **outcomes** in the environment in ways that will affect the performance measure of the agent in a nontrivial way. **Conscious thoughts don't create or affect reality**, but **only the actual actions matter** in that they can affect the environment and the results produced, independent of whatever reasoning led the agent to choose the particular action. Making the best action for mistaken or wrong reasons doesn't change the fact that it was still the best action! (Of course, rational thought should somehow correspond to rational actions in that compared to other forms of thought, rational thought would be more likely to produce rational actions.)
- The performance measure is set for the agent by its designer, to maximize the goals of the designer. The agent cannot change its performance measure, so it cannot trivially "win" by simply defining that whatever the current state of affairs, that is the best possible outcome.
- Rationality is not the same as **omniscience**: the agent should choose the action with the highest expected value based on its knowledge at the time. If some other action turns out to be better as the true complete state of the environment happens to be, you should not be a **result merchant** about it.

Once the environment is sufficiently complex in its actions and their consequences, an agent that chooses good or optimal actions seems to have an internal life of some kind when viewed from the outside.

Optimal actions can evolve without any conscious thought or cognitive model of the environment from the part of the entity that executes these actions. For this reason, science very rarely proves that some idea held by humanity is wildly inaccurate (and this is especially true for ideas that have important life and death consequences and get tested on a daily basis), but merely helps us understand the underlying reason why something that humanity thought to be true is actually true. We usually know that something is true (at least in the sense that we act as if that something were true whenever our decisions and actions have important consequences) long before we can explain *why* that something is true.

Properties of semantics

Bivalence: *Every sentence is either true or false in the given world*. Note that in this level of abstraction, “true” and “false” are merely words, and we could just as well use the words “foo” and “bar” (or “zero” and “one”). Holds for ordinary propositional logic and predicate logic, but not for three- or multivalued logics of uncertainty. In Bayesian probability calculus of propositions, a formula itself is true or false in the given world, it is just that our subjective knowledge is expressed as a number in the range [0, 1]. In fuzzy logic, a proposition is true to a degree that is in [0, 1] independently of what anybody thinks about it, and cannot be binarized by any amount of close examination.

Truth functionality: *The truth value of a sentence is solely determined by the truth values of its parts and the operator used to combine these parts*. Therefore the truth value of a sentence does not change if some part is rewritten as another part that is logically equivalent. Holds for ordinary propositional logic (and it follows that using n proposition symbols, there are exactly 2^n equivalence classes of essentially different logical formulas) and predicate logic.

However, truth functionality does not hold for **probabilities** (even if you know for a fact that P(A) = 0.2 and P(B) = 0.3, you still don't know the value of P(A and B) except for its trivial upper and lower bounds given by the Kolmogorov axioms). Nor does it hold for higher order logics that use **modal operators** such as **knowledge** or **belief** to nest logical sentences inside larger sentences. (The canonical example is how the sentences "Lois Lane knows that Superman can fly" and "Clark Kent is Superman" together do not entail the sentence "Lois Lane knows that Clark Kent can fly.")

Law of Excluded Middle: *Any sentence of the form* ***"A or not-A"*** *is true in every world*. Holds for propositional and predicate logic, but **this is not the same thing as bivalence**, since some more complex form of logic can well be three-valued with a third possible truth value "undetermined" and still satisfy LoEM. Does not hold for multivalued logics such as probabilities or **fuzzy logic**.

Properties of the inference engine

Soundness: *The inference engine produces only sentences that are entailed by the knowledge base*, that is, are guaranteed to be true in each world in which the knowledge base is true.

Monotonicity: *Adding more sentences to the knowledge base can never decrease the set of sentences that can be inferred from it*. Holds for resolution reasoning, but does not hold for mechanisms of default logic whose inferences may have to be retracted when new information arises.

Locality: *Inferences based on sentences can be performed independently of what other sentences are currently stored in the knowledge base.* Holds for resolution reasoning, but does not hold for default logic or probabilities, where all the available evidence must be considered. Strong independence assumptions are used to achieve locality in Bayesian belief networks.

Detachment: *Inferences based on sentences do not depend on the mechanism that originally produced those sentences*. The sentence itself is true or false in the given world, independent of which particular sound proof method you happened to use to establish your knowledge of it being true.

Principle of explosion: *From a contradiction, anything follows*. A system that contains both a sentence and its negation can be used to infer any sentence whatsoever. Holds for resolution refutation in propositional and predicate logics.

Objects, terms and functions in FOL

For simplicity, assume one constant symbol *bob*, one function symbol *s*, and the following axioms:

1. $\forall x : \neg(s(x) = bob)$
2. $\forall x: \forall y: (s(x) = s(y) \Rightarrow x = y)$

Intuitively, (1) says that Bob is nobody's successor, and (2) says that two objects have the same successor, then they are the same object. (Conversely, no two different objects have the same successor.) It follows from (1) and (2) that an infinite chain of objects emanates from *bob* using the function *s*.

Consider the three worlds W1, W2 and W3, all models of these axioms with this infinite chain. The difference being that W1 contains nothing else than that chain. W2 contains also as a *s*-cycle of three objects that is separate from this chain. W3 contains also a chain of objects that is infinite in both directions.

For each of these worlds, can you think up a FOL formula that is true in that world, but not true in the other two? It follows that these formulas or their negations are not entailed by the axioms (1) and (2), but can be true or false in worlds where (1) and (2) are true. However, note that the distinguishing formulas talk about the existence of some object that does not have a name in the language, since the infinite chain already uses up all the names that can be created from the symbols *bob* and *s*.

Nilsson's Elephants

Sam, Clyde and Oscar are elephants. Sam is pink, Clyde is gray. Clyde likes Oscar, and Oscar likes Sam. Prove that some gray elephant likes some pink elephant.

$\exists x: \exists y: \text{Gray}(x) \wedge \text{Pink}(y) \wedge \text{Likes}(x, y)$

Negating this claim gives us the formula that, conveniently enough, is already in 3-CNF:

$\forall x: \forall y: \text{not-Gray}(x) \vee \text{Not-Pink}(y) \vee \text{not-Likes}(x, y)$

1. Pink(sam)
2. Gray(clyde)
3. Likes(clyde, oscar)
4. Pink(oscar) or Gray(oscar)
5. not-Pink(oscar) or not-Gray(oscar)
6. Likes(oscar, sam)
7. not-Gray(X) or not-Pink(Y) or not-Likes(X, Y) (negation of claim we want to prove)
8. not-Pink(Y) or not-Likes(clyde, Y) (from 2 and 7 with X/clyde)
9. not-Pink(oscar) (from 3 and 8 with Y/oscar)
10. Gray(oscar) (from 4 and 9)
11. not-Pink(Y) or not-Likes(oscar, Y) (from 7 and 10 with X/oscar)
12. not-Pink(sam) (from 6 and 11 with Y/sam)
13. (empty) (from 1 and 12)

Removing literals and functions from FOL axioms

It is always possible to convert any FOL theory to an equivalent one that has no literals or function symbols but uses only relation symbols, by realizing that functions of n parameters are a special case of relations between $n + 1$ arguments. To remove some unary function $f(x)$ from the theory, create a new predicate $F(x, y)$ and add the axioms

1. $\forall x\colon \forall y\colon \forall z\colon F(x, y) \wedge F(x, z) \Rightarrow y = z$
2. $\forall x\colon \exists\ y\colon F(x, y)$

Then, all formulas P where the term $f(x)$ appears for some instantiation, convert them to form

3. $\forall y\colon F(x, y) \Rightarrow P'$

where P' is the formula constructed from replacing each occurrence of $f(x)$ by variable y.

First order logic + arithmetic is algorithmically unsolvable

As proven by Kurt Gödel in the thirties, there exists an algorithm that can produce all formulas that are entailed by the given set of FOL formulas. However, this proof was non-constructive, and the resolution refutation + unification algorithm was not discovered until the sixties.

In propositional logic using n propositions, there exist only 3^n possible clauses. (Each proposition can be in clause positive, in clause negative, or not be in the clause.) Therefore for propositional logic, resolution refutation will eventually either produce the empty clause, or reach the state where no more additional clauses can be produced by resolution.

However, for predicate logic, there exists an infinite number of terms even if only one function symbol f and one constant symbol is used, and therefore an infinite number of clauses. A systematic generation of clauses will eventually produce the empty clause if one can be produced, but there is no general way to determine how long this will take. If no empty clause can be produced, the generation of new clauses will continue forever.

When integers and their basic arithmetic is defined axiomatically in first order logic with Peano axioms, the resulting system has multiple models. That vague idea that we normally have in mind when we talk about integers is called the "standard model" of integers, but one can easily construct nonstandard models that contain our ordinary integers plus a whole bunch of others for which we don't have names for in our language of arithmetic. Some formulas are true in these nonstandard models and false in others, and so these formulas cannot be entailed by any first-order axiomatization of integers.

With the integers and their basic operations defined, concepts such as primality are easy to define:

$\forall x$: Prime(x) $\Leftrightarrow$ ($x > 1 \wedge \forall y : \forall z : x * y = z \Rightarrow (y = 1 \vee z = 1)$)

And yet nobody knows whether the following "Twin Prime Conjecture" is true or false in the standard model of integers. (Also note how we can cleverly say in first order logic that there exist infinitely many twin primes by saying that for any integer, there exists some larger integer that is a twin prime, that is, twin prime numbers exist without an upper bound.)

$\forall x : \exists y : y > x \wedge$ Prime(y) $\wedge$ Prime($y + 2$)

If there existed some algorithmic technique to determine whether this simple formula is true in the standard model of integers, or whether it is entailed by whatever logical axioms you use to define integers and their operations, surely some mathematician would have already used that technique to determine whether the above Twin prime conjecture is true or false! In fact, Tarski's Undefinability Theorem proves that it is generally impossible to define arithmetic truth using integer arithmetic. Since integer arithmetic encompasses computation, this proves that there is no general algorithm to determine whether the given arithmetic formula is true in the standard model of integers.

Godel's First Incompleteness Theorem proves that there are formulas that are true in this standard model of integers, but there cannot exist a formal proof that they are true.

Removing multiplication from integer arithmetic gives us much weaker (but still computationally prohibitively expensive) Presburger arithmetic that is consistent, complete or decidable: for any syntactically legal sentence, either that sentence or its negation is entailed by the axioms.

Default logic

The greatest strength of first order predicate logic is at the same time its greatest weakness: it is impossible to model any real world phenomenon with the requisite perfect accuracy so that interesting consequences can still be soundly inferred. Consider the following facts, expressed as a tiny story in spirit of the AI researcher Roger Schank. (The instructor happened to read one of his translated books, most likely "The Connoisseur's Guide to the Mind", in the local library, and here we are now.)

"*During her lunch break from her job at the bank, Linda walked into the greasy spoon diner next door. As she sat down, the waitress brought her the menu and a cup of coffee.*"

None of the following claims are logically entailed by this story, and yet it seems somehow more reasonable to assume the following claims as reasonable **defaults** rather than their negations, based on both of our experiences of how the world and society work in general, and the **social conventions of storytelling** that require that all salient facts are expressed up front to the reader.

- Linda's job at the bank does not involve operating heavy machinery or giving out prescription medication.

- Linda sat down on a chair at one of the customer tables (as opposed to, say, on the table itself, on the floor or behind the checkout counter).
- Linda took off her hat, gloves and coat before sitting down, but she did not take off her shoes.
- The table that Linda sat down at had napkins, ketchup and mustard somewhere on it. However, Linda did not pour any of this ketchup or mustard in her cup of coffee, nor did she eat the napkins.
- The coffee was hot, and Linda drank it directly from the cup instead of pouring it on her food. On the other hand, mustard and ketchup, if Linda later consumed any, were poured on the food.
- Linda is a human (as opposed to, for example, a service dog) and has at most two arms, at most two legs and exactly one head.

In ordinary predicate logic, if the axioms do not entail either formula *F* or not-*F*, we cannot soundly infer either one, since either one could be true in the world that we are currently in. But in many situations, it seems absurd to give both formulas equal credence. **Default logic** is a **nonmonotonic** generalization of predicate logic in which inference rules are extended to allow **default rules that are assumed to be true until they can be explicitly proven to be false**. For example, to say that every bird is assumed to be able to fly until explicitly proven otherwise,

```
Bird(x) : not-CanFly(x)
---------------------------
CanFly(x)
```

Default logic may require retracting previous inferences (and other inferences made from them) when new information arrives. It therefore does not allow **detachment** of truths from their proofs, unless the proof and the proofs of its premises do not use any default rules. Suppose your knowledge base has the formula

```
Penguin(x) => not-CanFly(x)
```

As long as you only have the axiom `Bird(tweety)`, you can infer `CanFly(tweety)`. Later, adding the observation `Penguin(tweety)` forces the retraction of this conclusion, and re-evaluation of all other facts that were derived using the premise `CanFly(tweety)`.

Knowledge base of default logic can still allow two contradictory conclusions to be inferred from the same axioms (the so-called "Nixon diamond" being the canonical example), in which case there has to exist some **meta-rule** to decide which conclusion is assumed to be true by default in such conflict. Even in the absence of such “second-order” contradictions, inference in default logic is clearly at least as difficult from the computational complexity standpoint as ordinary logic, for both predicate and propositional versions.

Furthermore, when combined with actions and their possible consequences of highly different severities, default rules can lead to absurd conclusions no matter which way you slice them. If you are going to take a long car trip, would your default assumption be that you will be involved in a car crash, or that you will

not be? It has to be one or the other, but if it is the former, the logical action is not to take the trip at all; and if it is the latter, the logical action is not to buy car insurance or even wear a seat belt! To be able to make rational decisions, we need probabilities to properly weigh the costs of each possible outcome.

Alternatively, we can force every sentence to be either true or false with logical meta-rules. Prolog and its negation operator use **negation as failure**, the principle that defines that a formula is false if and only if it cannot be proven to be true from the knowledge base. The related **closed-world assumption** states that only those things are true in the current world that can be proven to be true.

Dutch Book example of inconsistent beliefs (AIMA)

Proposition	Joe's belief	So we offer the bet	Joe's stake to our stake
A	0.4	A	6 to 4
B	0.3	B	7 to 3
A or B	0.8	not(A or B)	8 to 2

Possible outcomes for Joe

Bet	A and B	A and not-B	not-A and B	not-A and not-B
A	-6	-6	4	4
B	-7	3	-7	3
not(A or B)	2	2	2	-8
Net for Joe	-11	-1	-1	-1

Monty Hall Problem as the cliched Bayesian example

Let A, B and C be the propositions that the prize is behind door A, B and C, respectively. Let MA, MB and MC be the propositions that Monty opens door A, B or C, respectively, after the player chooses the door A.

Choose the door A. Monty opens door B, revealing a goat, therefore P(B | MB) = 0. Knowing this, how much are P(A | MB) and P(C | MB)?

P(A | MB)
= P(MB | A) P(A) / P(MB)
= P(MB | A) P(A) / (P(MB and A) + P(MB and not-A))
= P(MB | A) P(A) / (P(MB | A) P(A) + P(MB | not-A) P(not-A))

= 1/2 * 1/3 / (1/2 * 1/3 + 1/2 * 2/3)
= 1/6 / (1/6 + 2/6) = 1/6 / (1/2) = 1/3

P(C | MB)
= P(MB | C) P(C) / P(MB)
= 1 * 1/3 / P(MB)
= 1/3 / 1/2 = 2/3

(Or easier, by subtracting P(C | MB) = 1 - P(A | MB) - P(B | MB) = 1 - 1/3 - 0 = 2/3.)

Variation of Monty Hall, sort of "argument from ignorance"

Adding new options or information can never decrease the value of any game, assuming that we are free to ignore these options and information, and the consequences and payouts for the original options are not affected. Conversely, removing information or options can never increase the value of any game.

So, choose the door and close your eyes and ears, going "la la laa, I can't hear you". Let Monty do his spiel. You don't even care what he says or does.

If you do not switch, you win if you initially chose the prize door. P(Win | not-Switch) = 1/3.

If you switch, you win if you initially chose either of the goat doors. P(Win | Switch) = 2/3.

Adding the option of looking at what Monty does cannot make you worse off in the game, since you can always merely ignore this look in your decision making. Therefore in the actual game, switching wins with probability that is at least 2/3.

Athens Taxi Hit and Run

Let B mean "Taxi is blue", and A mean "Taxi appeared blue". We are given:

- P(B) = 0.1
- P(A | B) = 0.75
- P(A | not-B) = 0.25

P(B | A) = P(A | B) P(B) / P(A)
= 0.75 * 0.1 / P(A) = 0.075 / P(A)
P(not-B | A) = P(A | not-B) P(not-B) / P(A)
= 0.25 * 0.9 / P(A) = 0.225 / P(A)

Since these have to add up to one, P(A) = P(A | B) P(B) + P(A | not-B) P(not-B) = 0.075 + 0.225, and so

P(B | A) = 0.075 / (0.075 + 0.225) = 1/4

Since there are many more green taxis than blue taxis, for every correctly identified blue taxi there are still three mistakenly identified green taxis.

Solving probability problems with frequencies

Since **frequencies** are basically unnormalized probabilities (that is, they don't need to add up to one), many probability calculations are easier to do with them. Just be careful to do it right! (Some material on probability actually explains Bayes theorem with a similar notion of "Bayesian waterfall".)

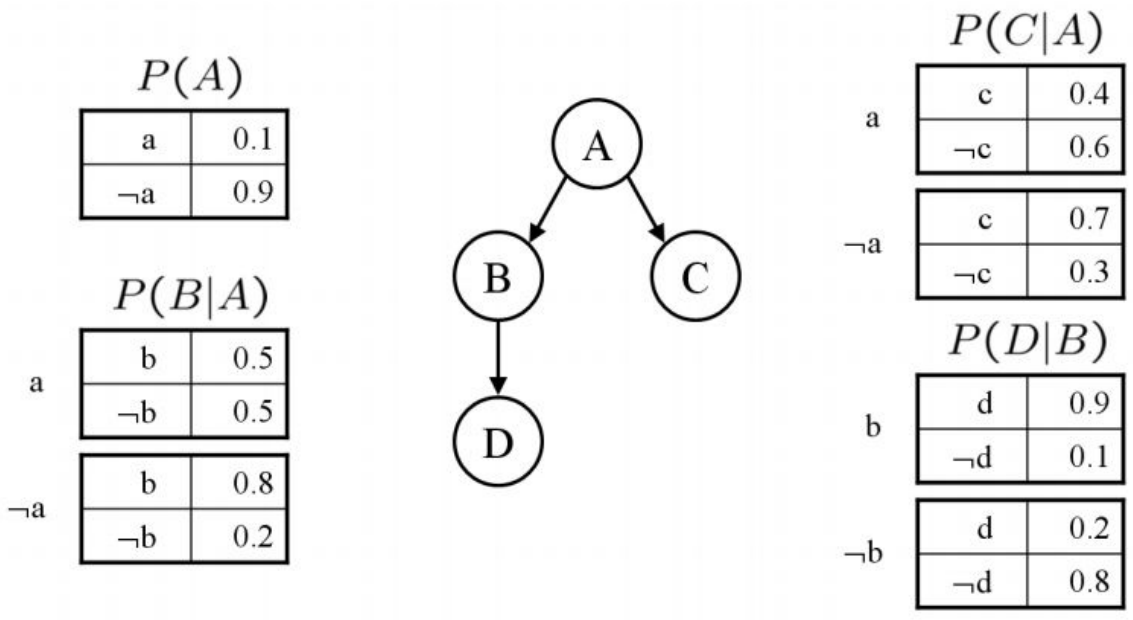
In some factory, 85% of produced items are good and 15% are defective. A good item is mistakenly rejected with probability 0.1, whereas a defective item is correctly rejected with probability 0.95. What is the probability that an item that the consumer buys from the store is defective?

Consider a batch of 10,000 items produced by the factory. Of these, 8500 are good and 1500 are defective. Of the 8500 good items, 850 were mistakenly rejected and 7650 reached the store. Of the 1500 defective items, 1425 were correctly rejected and 75 reached the store. Therefore from this batch, 7650 good items and 75 bad items reach the store. A consumer who buys a random item therefore gets a defective item with probability

75 / (75 + 7650) = 0.0097

That is, less than one percent.

Inference using a Bayesian network



(a) P(B and C | A)
= P(B | A) P(C | B and A)
= P(B |A) P(C | A) (*C is conditionally independent of B, given A*)
= 0.5 * 0.4 = 0.2

(b) P(D | A)
= P(D and B | A) + P(D and not-B | A) (*marginalization on B*)
= P(B | A) P(D | A and B) + P(not-B | A) P(D | A and not-B)
= P(B | A) P(D | B) + P(not-B | A) P(D | not-B) (*D is conditionally independent of A, given B*)
= 0.5 * 0.9 + 0.5 * 0.2 = 0.55

(c) P(A | D) = P(D | A) P(A) / P(D) (*straight up Bayes when going up*)
= (P(D and B | A) + P(D and not-B | A) P(A) / P(D) (*marginalization on B*)
= (P(B | A) P(D | B) + P(not-B | A)P(D | not-B)) P(A) / P(D)
= 0.055 / P(D)
P(not-A | D) = … = 0.684 / P(D) (*same way as P(A | D)*)

P(A | D) = 0.055 / (0.055 + 0.684) = 0.074 (since P(A | D) + P(not-A | D) = 1)

Nash equilibrium for two player simultaneous games

Zero sum game for two players, results given for maximizer A:

A\B	B1	B2	B3
A1	-2	6	5
A2	4	-1	-2

First, notice that B3 dominates B2 (from the point of view of B), so action B2 can be ignored.

	B1	B3
A1	-2	5
A2	4	-2

Principle of indifference: To find the Nash equilibrium for the above game, A should choose between his possible actions with probabilities that produce EV(B1) = EV(B3), making him indifferent to how B chooses his actions.

If action A1 is chosen with probability p and action A2 with probability $1 - p$, we get

EV(B1) = $2p - 4(1 - p) = 2p - 4 + 4p = 6p - 4$
EV(B3) = $-5p + 2(1 - p) = -5p + 2 - 2p = -7p + 2$

Setting these values to be equal we get

$6p - 4 = -7p + 2$
$13p = 6$
$p = 6/13$

In the long run, A should play A1 six times out of thirteen, and play A2 seven times out of thirteen.

If action B1 is chosen with probability q and action B2 with probability $1 - q$, we get

$EV(A1) = -2q + 5(1 - q) = -2q + 5 - 5q = -7q + 5$
$EV(A2) = 4q - 2(1 - q) = 4q - 2 + 2q = 6q - 2$

Setting these values equal we get

$-7q + 5 = 6q - 2$
$-13q = -7$
$q = 7/13$

In general, if there are n possible actions, there are $n - 1$ separate probabilities to solve from the set of linear equations given by the indifference equations.

Nash equilibrium for games with a saddle point

	B1	B2
A1	0	1
A2	-2	0

If an entry is a **saddle point**, that is, simultaneously the row minimum and column maximum, such as A1/B1 in the above table, the **pure strategy** where both players play that action with probability 1 is a Nash equilibrium for that game.

If we do the previous calculation for the above payoff matrix, we get $p = 2$, that is, A should choose action A1 with probability 2. Since the maximum possible probability is 1, this means that A will choose action A1 all the time, the difference reflecting the guaranteed severity of the mistake of choosing some other action.

Utility Theory

Agent has preferences about the possible outcomes of its decisions. These preferences are not rational or irrational, they simply are, depending on the design of the agent. For an artificial agent, the creator of that agent builds the agent to follow its creator's preferences. Rational actions maximize expected outcome.

Consider three possible non-numerical outcomes:

A: A duck
B: A chicken
C: Kick in the butt

For any two outcomes, agent can either prefer one to the other, denoted by $A > B$, or be indifferent between them, denoted by $A \sim B$.

Note that preferences are **ordinal**: if we know that $A > B > C$, we still can't say **how much** that agent prefers a duck to a chicken, or whether the agent would prefer two ducks to three chickens, etc.

In interesting problems, outcomes are uncertain. A **lottery** is a probability distribution of possible outcomes. If there are at least two possible outcomes, there exists an infinite number of lotteries, e.g.

[0.3, A; 0.6, B; 0.1, C]

Certain outcomes are special cases of lotteries with one outcome having probability one.

For the question of rational action to be meaningful, agents must have a preference ordering for all lotteries. For these preferences to be **coherent**, they must satisfy

Orderability: For any two lotteries A and B, exactly one of the three possibilities $A > B$, $B > A$ or $A \sim B$. Make up your mind, agent.

Transitivity: If $A > B$ and $B > C$, then $A > C$. Otherwise, you can be enticed into an infinite cycle of negative value trades.

Continuity: If $A > B$ and $B > C$, then there exists some probability p so that $[p, A; 1 - p, C] \sim B$. This quantity can be used to measure how much more the agent prefers A to B, relative to how much it prefers B to C.

Substitutability: If $A \sim B$, then in any lottery, A can be replaced by B and vice versa, with the agent being indifferent between the resulting lottery and the original.

Monotonicity: If $A > B$ and $p > q$, then $[p, A; 1 - p, B] > [q, A; 1 - q, B]$.

Decomposability: Nested lotteries can be expanded into single-level lotteries, with the agent being indifferent between the two lotteries. No fun in gambling.

If the agent's preferences satisfy these six, there necessarily exists a utility function *U* that maps outcomes to real numbers so that

If A > B, then U(A) > U(B).
If A ∼ B, then U(A) = U(B).
For any lottery, its utility is the weighted sum of the utilities of possible outcomes, weighted by their probabilities.

To maximize the expected outcome, it is necessary and sufficient to maximize U. This marries preferences to probabilities, giving **decision theory** as their synthesis.

It is important that U is not unique: shifting and scaling does not change the behaviour of the agent that acts to maximize its expected utility. Again, only the actions and their consequences matter, not the internal thoughts of the agent performing these actions.

This is handy in utility calculations, since we can shift the utility function so that some problematic term U(X) becomes zero and vanishes from calculations.

Lotteries can be used to determine the relative preferences of agents. If Alex and Bob both say that they believe that A > B > C, we can't pop open their heads like the hood of a car to look inside of how strong these preferences are. However, if the **revealed preferences** indicate that for Alex, B ∼ [0.2, A ; 0.8, C] and for Bob B ∼ [0.5, A ; 0.5, C], this fact reveals that Alex likes A relatively more than Bob compared to B and C. If both Alex and Bob currently have B, Alex is willing to accept a smaller chance to get A and become better off while accepting a bigger chance to get C and become worse off. Therefore, A is more valuable to Alex than it is to Bob.

Utility of Money

Agents that play for money have a **monotonic utility** for money so that if $x > y$, then $U(x) > U(y)$. However, since most people are **risk averse**, this utility is usually sublinear so that $U(2x) < 2\,U(x)$. This has important consequences in decision theory once the amounts in question become sufficiently large so that an action can have a positive expected value but negative expected utility, or vice versa.

Which one of the following would you choose, if a friendly billionaire (perhaps Tony Stark) entered the lecture room and offered either one to you right now?

A: [0.8, $4000; 0.2, $0]
B: [1.0, $3000]

Note that

EV(A) = $3200
EV(B) = $3000

However, due to the sublinear growth of utility of money, it may still well be that

U($3000) > U([0.8, $4000; 0.2, $0]) = 0.8 U($4000) + 0.2 U($0)

Therefore, most people would choose B instead of A. However, were the same choice offered to some professional billionaire options trader (for whom utility of money is effectively linear for a measly couple thousands of dollars), he surely would not hesitate to choose A.

The **certainty equivalent** of a lottery is the fixed amount so that the agent would be indifferent between the lottery and that certain amount. (What is the certainty equivalent of A?)

If CE(A) < EV(A), the agent would be willing to pay up to EV(A) - CE(A), the **insurance premium** of A, to exchange that lottery for certainty. This makes the insurance **industry** possible, especially in cases where the lottery consists of [*p*, status quo; 1 - *p*, huge disaster], with *p* being close to 1. Insurance companies, with billions in assets, can be considered to have a linear utility curve for the amounts of money equivalent to the price of a typical home, car or human life. Insurance industry therefore provides value to its customers, proportional to the difference between the insurance premium and the cost of insurance, which in a competitive insurance market becomes equal to the expected value of the insurance.

(By the way, don't proclaim that life has infinite value. If you do, then anything with the slightest chance of death has an infinite cost, making even the everyday actions of life impossible. Furthermore, there is no difference between the lotteries [1.0, kill one person] and [0.1, kill everyone on Earth]... or for that matter, the lottery [1.0, kill everyone on Earth], since each of these lotteries is assigned the same infinite value.)

Allais paradox

Choose between the two lotteries that this time do not guarantee any money, but just a chance:

C: [0.2, $4000; 0.8, $0], with EV of $800
D: [0.25, $3000; 0.75, $0], with EV of $750

Compare your answer to choice between A and B. Most people choose B over A, and C over D. That is, U(B) > U(A) and U(D) < U(C). Let us expand:

U(A) = U([0.8, $4000, 0.2, $0]) = 0.8 U($4000) + 0.2 U($0)
U(B) = U($3000)
U(C) = U([0.2, $4000; 0.8, $0]) = 0.2 U($4000) + 0.8 U($0)
U(D) = U([0.25, $3000; 0.75, $0]) = 0.25 U($3000) + 0.75 U($0)

As shown by the revealed preferences,

$0.8U(\$4000) + 0.2U(\$0) < U(\$3000)$
$0.2U(\$4000) + 0.8U(\$0) > 0.25U(\$3000) + 0.75U(\$0)$

The system has three unknowns U($4000), U($3000) and U($0) and two inequalities. However, using the trick of shifting the utility function so that we make U($0) = 0, we get rid of the term U($0) and now

$0.8U(\$4000) < U(\$3000)$
$0.2U(\$4000) > 0.25U(\$3000)$

If the second inequality is multiplied by 4, we get

$0.8U(\$4000) > U(\$3000)$

Whoops! The exact opposite of the first inequality. Something is clearly off with our preferences, but exactly what is it? (Answer: the fact that both private and social regret of **"resulting"** from losing decisions are not included in the calculation.)

Money in single and multiagent decisions

Fundamental question of Economics 101: Since we again cannot pop open people's brains like the hood of a car to examine the strength of their preferences and convictions, how can scarce resources be optimally distributed among multiple agents?

Example: three friends rent a shared apartment with three bedrooms A, B and C, which are not identical. How to decide who gets to live in which room, if all three have the same chain of preferences A > B > C ? We can't measure how many "utons" of pleasure each agent would derive from each room, all three enthusiastically claiming to be the one who would derive the largest number of utons from assigned space around him.

Solution: auction the bedrooms between the three as a percentage of rent paid. This converts the preferences to monetary amounts to measure them. (This assumes that all three friends have roughly similar utility curves for money, presumably true in this small scenario.)

In a large society of millions or billions of people, each one seeing a very limited part of the total, how can useful knowledge about important questions be produced when the agents don't even know about each other's existence?

Prediction markets offer options on propositions about future events with a definite deadline, paying some fixed amount (for simplicity, $1) if the proposition is true. The market price of the option reflects the probability that the proposition is true.

People have different knowledge, each one of us experiencing, seeing, knowing only a small part of the whole. The equilibrium price of the option is the aggregate of this knowledge, even though these participants don't even know of each other's existence!

More generally, ever since the days of the Silk Road, **price signals** are the most important tool that society uses to self-organize and solve the **economic allocation problem**. As in the roommate example, free market prices measure **honestly the underlying reality of preferences and wants**. Trying to control the prices is akin to trying to change the outside temperature by shaking the thermometer.

Experiences of centrally planned economies show the futility of trying to solve the economic allocation problem without information given by free market prices. Despite the narcissistic beliefs of the advocates of such systems, even being the smartest, wisest, most moral and the best person in every humanly possible way would not magically confer that person the superpowers of the necessary omniscience in economic allocation to the would-be central planner.

The classic question to illustrate this is to make you a central planner of an economy without free market prices. In this exercise, you just have to decide whether the given small plot of land is used to produce strawberries or raspberries, so we ignore all possible other uses for that land in this toy problem. Now pray tell, how do you decide which type of berry is going to produce on that land, and how does your decision process differ from flipping a coin?

Bayesian updating: is this coin weighted?

If some repeated Markovian event *A* has a frequentist probability of 0.5, many people tend to subscribe to "gambler's fallacy" and mistakenly believe that in the long run, the difference between how many times *A* and not-*A* occur pulls towards zero, as if there were some invisible magical rubber band anchored at zero that makes *A* more probable if is has occurred fewer times than not-*A*, and vice versa. For example, if a roulette wheel has just given "red" six times in a row, many people believe that "black" is now somehow "due" making its probability greater than 0.5 and thus a positive expectation gamble.

This is, of course, rubbish: a Markovian random walk has no fixed anchor to "remember" where the walk started from. In the long run, the **proportion** of *A* and not-*A* will approach 0.5 (that is the very definition of frequentist probability, after all), but their **absolute difference will swing and grow both ways without any bound**. Proper randomness, in general, behaves very differently from how most people imagine it to behave. Unaided by dice, coins and other physical tools but forced to work eyes closed with only their minds, most people could not begin to emit results that would pass even the most rudimentary statistical tests of randomness. (The reader might think up **ensemble** methods to combine such bad RNG's into a better RNG, analogous to boosting the performance of weak learners into one more accurate learner.)

However, every fallacy always has its second-order "fallacy fallacy", and the gambler's fallacy is no exception to this human tendency. Suppose somebody ignores all these "unknown unknowns" and

proclaims that even if some coin has given heads a hundred times in a row, it still has the same probability of one half to produce tails the next throw. Such reasoning is valid only in some microworlds where the proverbial spherical cows run in a vacuum, hermetically sealed from the rest of reality. In the real world, at some point we must start assigning some nonzero probability to the fact that the coin might not be entirely fair, but is actually a weighted trick coin. (These are known to exist in our world, so their prior probability is greater than zero.)

In the spirit of the course slides example of five bags of candy with different proportions of cherry and lime, consider two hypotheses:

H_1: The coin is a normal fair coin. $P(H_1) = 1 - p$, with p very small.
H_2: The coin is a weighted trick coin that always produces heads. $P(H_2) = p$.

How many heads do we need to see in a row until H_2 becomes as likely as H_1?

Let X_n mean that we have flipped the coin n times, and it came up heads each time. Bayes rule:

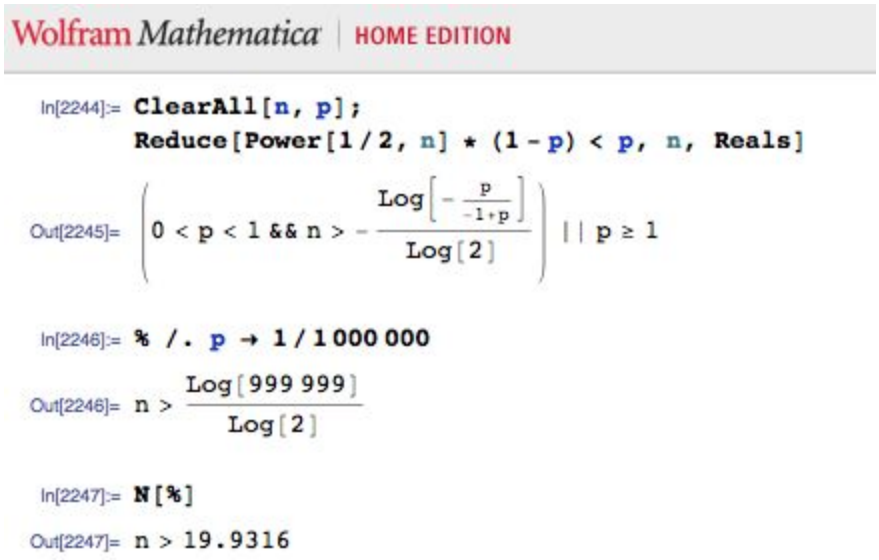
$P(H_1 \mid X_n) = P(X_n \mid H_1)\, P(H_1) / P(X_n)$
$P(H_2 \mid X_n) = P(X_n \mid H_2)\, P(H_2) / P(X_n)$

Since both expressions use the same normalizing factor $P(X_n)$, we can drop it when we compare the values to see which one is greater. Substituting what is known, we get

$P(H_1 \mid X_n) = 0.5^n * (1 - p)$
$P(H_2 \mid X_n) = 1 * p = p$

$0.5^n * (1 - p) < p$
$0.5^n < p / (1 - p)$
$n \log 0.5 < \log(p) - \log(1 - p)$ (let's use base 2 logarithms for convenience)
$-n < \log(p) - \log(1 - p)$
$n > \log(1 - p) - \log(p)$
$n > \log((1 - p) / p)$

Let's plug in some values of p to try it out. Suppose one coin in million is a trick coin:



REM Process (Range, Equity, Maximize)

Canonical example application of the REM process would be various forms of poker, but the general principle is universal for all situations where you must choose the optimal action from your palette of possible actions against unknown nature or adversary.

Heads-up Texas Hold'em Poker, as if it were the year 2007 again and poker is hotter than Hades. Having flopped bottom two pair, we bet half pot on the flop and turn, the opponent "Crafty Joe" calling each time. The river card does not pair the board nor create a possible flush, but does create a possible straight. Joe chortles and bets the pot, putting himself all in with the last $50 in his chip stack. The pot currently stands at $100, with $50 to us to call or fold.

Range: Infer a probability distribution of the possible states that the environment and the opponent can be in, using Bayesian reasoning based on the previous actions and the opponent model. Say:

Joe is snowing with air: 30%
Joe has pair or weaker two pair: 20%
Joe has better two pair: 20%
Joe made his straight: 20%
Joe had small pocket pair and made (or was slowplaying from flop) three of a kind: 10%

Equity: Compute the expected value of each possible action as a sum of values of that action in each possible state, weighted by the probabilities of the states. (Be careful of the **sunk cost fallacy**. Once the money goes in the pot, it is no longer your money.)

EV(Fold) = $0
EV(Call) = (0.3 + 0.2) * $100 + (0.2 + 0.2 + 0.1) * (-$50) = 0.5 * $100 + 0.5 * (-$50) = $25

Maximize: Choose the action with the highest EV. Verdict: call.

Opponent modeling is important, since against "Ronald Rock" or "Ned Newbie" whose calling and betting tendencies would tend to be very different from those of Crafty Joe, the world state probabilities and therefore all possible action values might also end up being very different! For example, Ned Newbie is not capable of complicated plays, so we would assign him a higher probability of having the range that his actions are showing.

Note also that the opponent modeling includes **second order modeling**, that is, modeling the opponent's internal model about our behaviour! For example, if we have previously given Joe an overall impression of being a wild bettor and bluffer, Joe might now rather tend to check his winning hands and wait for us to bet. (Reputedly, high level professional poker includes thinking up to third and fourth level in this sense.)

Also, if we change the situation so that Joe still has $200 left in his stack so that we could reraise him, should we reraise even if we were 90% sure that we have a better hand? No way! This reraise can only gain if it either **makes Joe fold a hand that is better than our hand, or call with a hand that is worse than our hand**. In this situation, both possibilities are negligible. Such a reraise therefore cannot really gain anything, but can lose a lot more in the situation where Joe was not bluffing and will happily call our reraise to accept our kind donation of more money than what he was properly entitled to in that situation.

Nash equilibrium bluffing probability in poker

Some suitable form of heads-up poker. Assume pot is currently $100, and you have $25 left in your stack. In the last draw, pocket card or river card, whatever way that form of poker has it, you either make the **best hand** or end up with a **busted hand**. Of course you know exactly what you got, but from the point of view of your opponent, you could have either one in this situation with the same 50% probability. (In poker and all card games, you need to maintain two states of mind: your own, and your opponent's simulated state of mind given the things that he knows from the actions that he has seen you do.) Your opponent checks to you.

If you make the best hand, surely you will bet (and win either $125 or $100, depending on whether your opponent calls or folds). But if you did not make your hand, with what probability p should you try to bluff with your last $25, instead of just checking and giving up the entire pot?

From your opponent's point of view, if you bet, your hand is bust with probability

P(Bust | Bet)
= P(Bet | Bust) P(Bust) / P(Bet)
= P(Bet | Bust) P(Bust) / (P(Bet and Bust) + P(Bet and not-Bust))
= P(Bet | Bust) P(Bust) / (P(Bet | Bust) P(Bust) + P(Bet | not-Bust) P(not-Bust))
= p * 0.5 / (p * 0.5 + 1 * 0.5)

$= p * 0.5 / ((1 + p) * 0.5)$
$= p / (1 + p)$

Indifference principle again gives us the Nash equilibrium from the rule that the expected value of calling and folding should be equal to the opponent.

EV(Call)
= P(Bust | Bet) * $125 + P(not-Bust | Bet) * (-$25)
$= p / (1 + p) * \$125 + (1 - p / (1 + p)) * (-\$25)$

EV(Fold) = $0

Solving for p in equation EV(Call) = EV(Fold) , we get $p = 1/5$. So in this situation, the game theoretic optimum is trying a bluff one time out of five. Hey, you might get lucky. Also, whatever happens, don't be a result merchant about it, since you played the hand correctly in the sense of expected value maximization, and that is all that matters in the long run. Even if the bluff fails this time, the next time you make a winning hand, your opponent is more inclined to call your bet.

Value of Perfect Information

Again, a game of heads up Texas Hold'Em poker. Also, again you have flopped two pairs, with two hearts on the flop and none in your hand. The opponent called your bets on the flop and turn. The river card put the third heart on the board, completing a possible flush. The opponent chortles how lucky he is and bets the pot, representing a made flush. This crafty guy might be bluffing, though, since we have seen him pull stunts like that many times before. Pot is again $100, with $50 for you to call or fold. (Again, raising is a pointless exercise in futility that cannot possibly gain and can only compound your losses, since your opponent would really only call this raise when he has the hand to beat you.) Given no other information, you still intend to call, since this guy bluffs with the right frequency when he missed.

Suppose that before you act, a magic genie appears and gives you the option to buy (with money taken from your wallet, separate from your chip stack) a peek at one of the opponent's hole cards. How much would it be rational to pay for such a peek? (The opponent keeps his cards in his hand randomly in either order, and the peek will randomly reveal one of them.) For the moment, let us ignore all ethical issues of taking such a peek outside the rules and boundaries of the game.

More generally, when the expected value of an action depends on a number of unknown propositions, how much would it be rational to find out the value of some particular proposition?

Surely we would not pay $100 to find out whether we can win $100. But equally surely, we would pay $1 in a heartbeat rather than folding our equity in the pot. So what is the maximum amount that is rational to pay?

Suppose that your peek reveals that the hole card **is not a heart**, so you know for a fact that you have the best hand. What exactly did you gain from your peek?

Suppose that your peek reveals that the hole card **is a heart**. What exactly did you gain from your peek? What further assumptions does this evaluation depend on?

Nothing beats the trusty old rock

```
In[91]:= ClearAll[pm, po, p1, p2, p3, q1, q2, q3];
         penalty = +2;
         pm = {{0, -penalty, +1}, {+penalty, 0, -1}, {-1, +1, 0}};
         poa = Transpose[pm].{q1, q2, q3};
         pob = pm.{p1, p2, p3};
         Solve[{q1*pob[[1]] == q2*pob[[2]] && q2*pob[[2]] == q3*pob[[3]] &&
            p1*poa[[1]] == p2*poa[[2]] && p2*poa[[2]] == p3*poa[[3]] && p3 == 1-p1-p2 &&
            q3 == 1-q1-q2 && 0 < p1 < 1 && 0 < p2 < 1 && 0 < p3 < 1 && 0 < q1 < 1 &&
            0 < q2 < 1 && 0 < q3 < 1}, {p1, p2, p3, q1, q2, q3}]
```

$$\text{Out[96]=}\ \left\{\left\{p1 \to \frac{1}{4},\ p2 \to \frac{1}{4},\ p3 \to \frac{1}{2},\ q1 \to \frac{1}{4},\ q2 \to \frac{1}{4},\ q3 \to \frac{1}{2}\right\}\right\}$$

```
In[109]:= ClearAll[pm, po, p1, p2, p3, q1, q2, q3];
          penalty = +1000;
          pm = {{0, -penalty, +1}, {+penalty, 0, -1}, {-1, +1, 0}};
          poa = Transpose[pm].{q1, q2, q3};
          pob = pm.{p1, p2, p3};
          Solve[{q1*pob[[1]] == q2*pob[[2]] && q2*pob[[2]] == q3*pob[[3]] &&
             p1*poa[[1]] == p2*poa[[2]] && p2*poa[[2]] == p3*poa[[3]] && p3 == 1-p1-p2 &&
             q3 == 1-q1-q2 && 0 < p1 < 1 && 0 < p2 < 1 && 0 < p3 < 1 && 0 < q1 < 1 &&
             0 < q2 < 1 && 0 < q3 < 1}, {p1, p2, p3, q1, q2, q3}]
```

$$\text{Out[114]=}\ \left\{\left\{p1 \to \frac{1}{1002},\ p2 \to \frac{1}{1002},\ p3 \to \frac{500}{501},\ q1 \to \frac{1}{1002},\ q2 \to \frac{1}{1002},\ q3 \to \frac{500}{501}\right\}\right\}$$

Utilities trading

```
In[116]:= ev = 1/50*10 + 1/2000000*1000000 - 1
```

$$\text{Out[116]=}\ -\frac{3}{10}$$

```
In[126]:= ClearAll[u0, um1, u10, umil];
          Reduce[u0 < (1-1/50-1/2000000) um1 + 1/50 u10 + 1/2000000 umil &&
            u10 == -10*um1 && u0 == 0 && um1 < u0 < u10 < umil, {umil}, Reals]
```

```
Out[127]= um1 < 0 && u10 == -10 um1 && u0 == 0 && umil > -40000 u10 - 1959999 um1
```

Snap off those bluffs!

```
In[126]:= ClearAll[u0, um1, u10, umil];
          Reduce[u0 < (1 - 1 / 50 - 1 / 2 000 000) um1 + 1 / 50 u10 + 1 / 2 000 000 umil &&
            u10 == -10 * um1 && u0 == 0 && um1 < u0 < u10 < umil, {umil}, Reals]

Out[127]= um1 < 0 && u10 == -10 um1 && u0 == 0 && umil > -40 000 u10 - 1 959 999 um1

In[147]:= ClearAll[p, q];
          pb = p * 35 / 45 + 10 / 45;
          sol = First[Solve[(10 / 45) / pb * -8 + (p * 35 / 45) / pb * 26 == 0, {p}]]
          pp = p /. sol;
          pbb = pb /. sol;
          Solve[q * (10 / 45) / pbb * -8 + (1 - q) * (pp * 35 / 45) / pbb * 26 == 0, {q}]
```

Out[149]= $\left\{p \to \frac{8}{91}\right\}$

Out[152]= $\left\{\left\{q \to \frac{1}{2}\right\}\right\}$

Bias, variance, and no free lunch theorem

The entire supervised learning problem depends on the assumption that **there exists something to learn**, so that it is possible to encode the **true hypothesis** that describes the laws of the problem domain using significantly fewer bits than are required to encode the training samples.

Consider the problem of creating a binary classifier for the problem domain that consists of a row of n boxes, each one containing one of the values *true* or *false*, of which you can freely choose any subset as your training data. However, these boxes have been filled with random coin flips independent for each box. Even if you knew the contents of any n - 1 boxes of your choice, this would not tell you anything about the contents of the remaining box.

(Also, elementary combinatorics proves that the probability that the random coin flips fill these boxes with truth values that can be compressed into a hypothesis that can be described using fewer than n bits is vanishingly small, since there exist 2^n possible different hypotheses for the truth value combinations, but only at most 2^m of these can be encoded in m bits.)

In general, an infinite number of possible hypotheses would fit the given training data equally well, so the decision of which hypothesis to choose depends on the **bias** of the algorithm, its tendency to converge to one particular hypothesis due to the way that the algorithm works.

To explain the **variance** of a learning algorithm, suppose we have ample training data and we split it randomly in two halves which are used to generate two independent hypotheses h_1 and h_2 using the same learning algorithm. Assuming that half of the training data was enough to construct a hypothesis, we

would expect $h_1(x)$ and $h_2(x)$ to agree most of the time when given a brand new instance x to classify. The variance of the learning algorithm measures how often this is not the case.

Consider the following two extreme cases learning algorithms to construct a binary classifier that might seem almost silly, due to essentially the same fallacy that causes people to believe that 1, 2, 3, 4, 5, 6 is somehow lower probability outcome of next week's draw of Lotto 6/49 than 4, 7, 12, 24, 29, 40.

Closed world: For any instance x outside training data, always return *false*.

Lazy coin: For any instance x outside training data, flip a coin (and memoize for possible future queries).

The closed world algorithm has the largest possible bias, but zero variance (for instances that were not used to train either hypothesis). The coin flip algorithm is the exact opposite in that it has zero bias (the coin flips could produce any classifier hypothesis whatsoever!), but the largest possible variance. All other learning algorithms fall somewhere between these two. In general, the more internal degrees of freedom some learning algorithm has, the more variance and less bias it exhibits. Increased variance also makes the algorithm more sensitive to noisy training data.

But why do those two learning algorithms seem somehow so very silly to us, as if by some cosmic sense of justice they have no right to solve any actual problem? Well, analogous to how one choice of lottery numbers is no better or worse than any other choice of lottery numbers (including the classic and highly counterintuitive 1, 2, 3, 4, 5, 6, 7), it turns out that when averaged over the space of all possible classification problems, **neither algorithm is better or worse than any other learning algorithm.** Consider two worlds W_1 and W_2 that are mirror images of each other in that they agree for classifications of training data D, but assign the exact opposite classifications for every other instance. Let h_c be the hypothesis that the Closed World algorithm produces for D, and let h be any other hypothesis whatsoever. By however much h beats h_c when measured over W_1, it will lose to h_c by that exact same amount in W_2.

Since h_c beats any hypothesis in exactly half the possible worlds and loses in the other half, it cannot be said to be better or worse unless we have some additional information other than the training data D about the world that we are in.

The same argument works just as well for any two hypotheses that are consistent with the training data. There is no free lunch in learning. As the more famous version of this theorem in optimization problems puts it, "**If an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems.**"

So perhaps the real problem in machine learning is choosing the learning algorithm whose inherent bias just happens to fit well into the underlying shape and structure of the problem domain!

Important terms in philosophy of artificial intelligence

The outline of previous versions of this course finished with the lecture about the philosophical issues of artificial intelligence. As interesting and thought-provoking that entire topic would be for discussions both serious and in jest, the virtualized schedule from Fall 2020 onwards uses this time for one more lecture on Prolog in the middle of the course. However, here is the original row of the course outline document with the links to the interesting keywords in both Wikipedia and a couple of other places.

List of Winograd Schemas
AI-complete problems (Moravec's paradox, AI effect, Commonsense reasoning)
Expert system (Dreyfus model of skill acquisition, Four stages of competence)
ELIZA (effect)
SHRDLU
Philosophy of AI (Ethics)
Physical symbol system, Church-Turing-Deutsch principle
Symbolic AI ("GOFAI", "neats and scruffies")
AI Winter (History of artificial intelligence, Timeline of artificial intelligence, Timeline of machine learning)
Turing test
Qualia (Mind-body problem, Functionalism, Dualism)
Chinese room
Intentional stance
Gödel, Escher, Bach (Strange loop)
Rationality from AI to Zombies