A probabilistic semantics for natural language

Jean-Philippe Bernardy and Aleksandre Maskharashvili

32nd European Summer School in Logic, Language and Information,
Week 2: 9–13 Aug 2021

Contents

1 Inference under uncertainty: Problem description **3**
- 1.1 Logic as the means of categorical judgments/inferences 4
  - 1.1.1 Aristotelian Syllogisms 4
  - 1.1.2 First-Order Predicate Logic 5
- 1.2 Inferences that are non-categorical (non-logical inferences) 6
- 1.3 A type of non-logical inference: Probabilistic inference 7
- 1.4 Linguistic phenomena involved in probabilistic inferences 7
  - 1.4.1 Generalized Quantifiers 8
  - 1.4.2 Adverbs of Frequency 8
  - 1.4.3 Graded Adjectives and Comparatives 9

2 Probability theory, Bayesian Reasoning, Probabilistic programming **9**
- 2.1 Motivational Problem (1) 9
- 2.2 Motivational Problem (2) 10
- 2.3 Elements of Probability theory 10
  - 2.3.1 Concept: Probability distribution 10
  - 2.3.2 Notation 11
  - 2.3.3 Dependent and Independent events and variables 11
  - 2.3.4 Examples 12
  - 2.3.5 Conditional probability (1) 12
  - 2.3.6 Conditional probability (2) 12
  - 2.3.7 Probability Laws 13
  - 2.3.8 Random variables with priors (discrete) 14
  - 2.3.9 Evidence and posteriors 16
- 2.4 Probabilistic Programs 16
  - 2.4.1 Probability distributions (1) 17
  - 2.4.2 Probability distributions (2) 17
  - 2.4.3 Constants 18
  - 2.4.4 Sequencing instructions 18
  - 2.4.5 Observations 18
  - 2.4.6 Expected truth value (aka “Probability”) 19

2.4.7 Example: drug test (Wikipedia) . . . 20
2.4.8 Answer To Introductory Problem . . . 20
2.4.9 Meaning of probabilistic programs . . . 21
2.4.10 Exercise: Evaluating the answer introductory problem . . . 24
2.4.11 Final note on Beta distribution. . . . 24
2.4.12 Exercise: the children problem . . . 25
2.4.13 Exercise: betting on games . . . 25

3 Compositional translation of inference problems into probabilistic programs 25
3.1 Montagovian semantics . . . 25
3.2 Interpreting inference problems: recipe. . . . 26
3.3 Interpretation of semantic categories . . . 27
3.3.1 Propositions . . . 27
3.3.2 Individuals . . . 27
3.3.3 Reminder: set cardinalities . . . 28
3.3.4 Space of predicates . . . 28
3.3.5 Idea 1 . . . 28
3.3.6 Idea 2: Boxes . . . 29
3.3.7 Idea 3: Your Idea! . . . 29
3.3.8 Common nouns . . . 29
3.4 Interpretation of semantic operators . . . 30
3.4.1 Generalized quantifiers . . . 30
3.4.2 Universal Quantifiers . . . 33
3.4.3 Existential Quantifiers . . . 34
3.4.4 Comparatives . . . 34
3.5 Subsective Graded Adjectives . . . 36
3.5.1 Reminder: intersective vs subsective adjectives . . . 36
3.5.2 Probabilistic interpretation of subsective adjectives . . . 36

4 Computations: MCMC, Gradients 37
4.1 Exact evaluation . . . 37
4.1.1 Example: drug test . . . 37
4.2 Continuous case . . . 37
4.3 Markov Chain Monte Carlo . . . 38
4.3.1 Monte Carlo methods: . . . 38
4.3.2 Markov Chain . . . 38
4.3.3 MCMC . . . 38
4.3.4 The Structure of the probabilistic program influences performance . . . 39
4.3.5 Inner evaluation of proportions . . . 39
4.4 More methods! . . . 39
4.5 Returning to our motivating example . . . 39
4.5.1 Choice of prior (again) . . . 40
4.6 More Probabilistic programming packages . . . 40

5 Test suite for Probabilistic Inference, Outlook **40**
5.1 Building a corpus of probabilistic inference 40
5.1.1 Generalized quantifiers 40
5.1.2 Bare plurals, Generics, and Indefinite Noun phrases 41
5.1.3 Adverbs of frequency 41
5.1.4 Adjectives and Comparatives derived from them 41
5.2 Evaluation criteria of a probabilistic inference system against the corpus 42
5.3 Probabilistic approach to pragmatics 42
5.3.1 The proviso problem 42
5.3.2 The Rational Speech Act (RSA) model 43

6 References **44**
6.1 Our references 45

Preamble: finding the learning material

- Clone the repository: `git clone https://github.com/jyp/ProbInfer; cd ProbInfer` (The lecture notes are in `ESSLLI.org`)

(If you're running MacOS, you may see `xcrun: error: invalid active developer path (/Library/Developer/CommandLineTools)`, in which case you need to run `xcode-select --install`)

- Install the dependencies using nix (see also `https://nixos.org/guides/install-nix.html`)

```
sh <(curl -L https://nixos.org/nix/install)
```

(If you're running MacOS, you may run into more hurdles. For example you will have to type your password, and if it's empty, it won't work.)

Then you can do

```
nix-shell
```

Within this shell, you can load examples and run them.

- Load an example: `ghci DiceExample.hs`
- run: `doit`

What this program does is explained in the second lecture!

1 Inference under uncertainty: Problem description

Link to the video lecture: `https://youtu.be/3zQVGohcVd4`

At least since Aristotle, reasoning with and within natural language has been subject of studies. Traditionally, many of these studies were focused on identifying a logical apparatus of human reasoning. More recently, various logic based frameworks where proposed to study various aspects of human reasoning, including formalizing

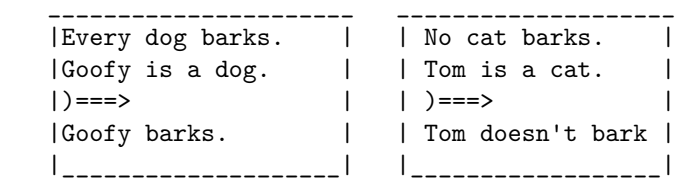
notions of knowledge, beliefs, hypothesis, making new conclusions based on that, updating knowledge base, etc.

Together with logical aspects, human linguistic activity as well involves reasoning with and about incomplete, uncertain, vague, implicit information. This includes making inferences, drawing conclusions with uncertain information at hand.

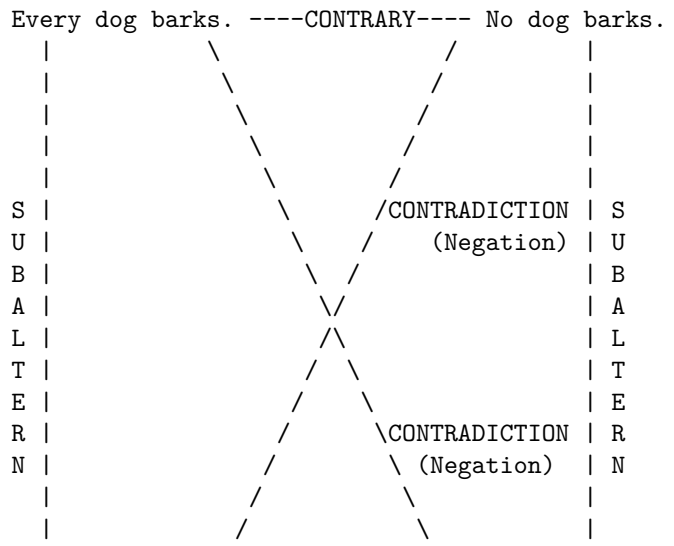
1.1 Logic as the means of categorical judgments/inferences

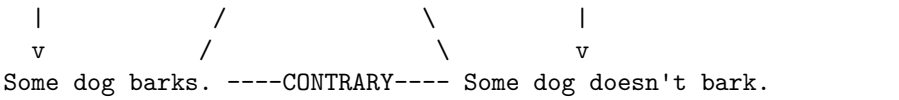
- Aristotelian Syllogisms
- limitations of Aristotelian Syllogisms for reasoning
- (see also "all men are mortal" below)

1.1.1 Aristotelian Syllogisms



What comes after the)===> symbol can be inferred from what is before. Propositions before the symbol)===> are called *premises*, and the statement after)===> is usually called *the conclusion* (note that there can be many or even no premise, but there is one and only one conclusion). We may say that the conclusion follows or is entailed from the premises. The premises together with the conclusion is called an *argument*.





We can do a bit more in the Aristotelian settings:

Every cat is a mammal.	No insect barks.
Every mammal eats.	Every fly is an insect.
)===>	)===>
Every cat eats.	No fly barks

In short, Aristotelian syllogisms concern with universally quantified statements, their subalterns, and negations.

1.1.2 First-Order Predicate Logic

In mathematical logics, First-Order Logic (FOL, Predicate Calculus) can be seen as a mathematical heir of the Aristotelian syllogisms. Introduced by Gottlob Frege in the end of 19th Century, FOL makes use of:

- logical connectives (&, ∨, ⟶, ¬)
- constants and variables (mary, john, 1, 0, 2, x, y, etc.)
- n-ary predicates (WOMAN, MAN, WALK, HUG, MEET, SEND, LOVE, etc.)
- n-ary functional symbols (+, -, etc.)
- quantifiers ∀x and ∃x, for every variable x
- A `term` is obtained by applying n-ary function to constants,

variables, and/or other terms, which are n in total.

- An `atomic formula` is an n-ary predicate whose all places are occupied by

terms, i.e., it is of shape P(t1,...,tn), where P is an n-ary predicate symbol and t1,...,tn are terms.

- A `literal (aka molecule)` is an atomic formula or its negation, that is,

it is of shape P(t1,...,tn) or ¬P(t1,...,tn) where P is an n-ary predicate symbol and t1,...,tn are terms.

- A `formula` is either:
  - a literal
  - application of logical connectives to formulae
  - application of a quantifier to a formula: ∀x(A) or ∃x(A)

Pluto is a pet	PET(pluto)
Goofy hugs Pluto	HUG(goofy, pluto)
Pluto is a dog and a star	DOG(pluto) & STAR(pluto)
If Goofy is late, Pluto barks	LATE(goofy) $\longrightarrow$ BARK(pluto)
Pluto is not mortal	$\neg$MORTAL(pluto)
Either Pluto or Goofy barked	BARK(pluto) $\vee$ BARK(goofy)
A man walks	$\exists$x(MAN(x) & WALK(x))
Every man walks	$\forall$x(MAN(x) $\longrightarrow$ WALK(x))
Every woman saw Joe	$\forall$x(WOMAN(x) $\longrightarrow$ SEE(x,joe))
Every woman saw a man	$\forall$x(WOMAN(x) $\longrightarrow$ $\exists$y(MAN(y) & SEE(x,y)))
	$\exists$y(MAN(y) & $\forall$x(WOMAN(x) $\longrightarrow$ SEE(x,y)))

Example of FOL usage:

```
____ENGLISH_____________________FOL_TRANSLATION__________
| Mary is a woman                WOMAN(mary)            |
| Mary walks                     WALK(mary)             |
| )===>                          )===>                  |
| A woman walks                   x(WOMAN(x) & WALK(x)) |
-------------------------------------------------------
```

How to prove $\exists$x(WOMAN(x) & WALK(x)) from the premises WOMAN(mary) and WALK(mary)? In FOL, it's quite natural to have the following rule: If A(k) holds, then $\exists$x(A(x)) also holds (where k is a term and x is not free in A(k)). By assuming WOMAN(mary) and WALK(mary) hold, then (WOMAN(mary) & WALK(mary)) holds as well. In this latter formula, we can apply the FOL rule for $\exists$, we get: $\exists$x(WOMAN(x) & WALK(x)). Q.E.D.

1.2 Inferences that are non-categorical (non-logical inferences)

Not all inferences are logical though. Consider the following inference problems:

Teddy just got home.	$\models$? Teddy is hungry.
Teddy just drank several glasses of milk.	$\models$? Teddy is not hungry.
Teddy loves cars.	$\models$? Teddy drives a car.
Teddy hugged a cat.	$\models$? Teddy got a cat.
Teddy enjoys listening to Led Zeppelin.	$\models$? Teddy loves English bands.
Teddy is 18 months old.	$\models$? Teddy weights around 15 kg.
Teddy can mimic dogs and cats.	$\models$? Teddy can speak Georgian.
	$\models$? Teddy is a bear.

One can imagine that the above examples, can be justified by using common-sense knowledge. Speakers, in many cases, may make extensive use of such knowledge in order to navigate within the context, make specific decisions and take respective steps.

The kind of inferences show above are rather hard to formalize in any classical deductive systems, because they require internalization a large amount of common sense knowledge.

Here, we limit our discussion to problems that do not require external knowledge about the world. And in many of the cases we study, we will be concerned with problems which are *similar* to the Aristotelian square, and generalizations thereof. We will use generalized quantifiers and inference problems built with them, which we call *probabilistic* inference problems.

1.3 A type of non-logical inference: Probabilistic inference

One of the main questions that we want to address is the following: Are the rules governing reasoning under uncertainty part of the same system as the logical rules? Irrespective of their ontological classification, those rules are part of human reasoning and are clearly expressed in natural languages (like English) with the specific morpho-syntactic and/or lexical means.

We call an inference demonstrative if the premises necessitate the conclusion. In other words, a demonstrative inference is any inference that is truth-preserving. For example, we can say that Aristotelian syllogism are only concerned with demonstrative inferences (given that premises are true, the conclusion is true).

If an inference is not demonstrative, we call it non-demonstrative. That is, the conclusion is not necessary to hold (be true) given that the premises hold. Hence, non-demonstrative inferences are those which are not truth-preserving. We are interested in non-demonstrative inferences. Of course, many "wrong" (unreasonable, rejectable) inferences are non-demonstrative.

"If, however, there is any kind of inference whose premises, although not necessitating the conclusion, do lend in weight, support it, or make it probable, then such inferences possess a certain kind of logical rectitude. It is not deductive validity, but it is important anyway." *Wesley C. Salmon, The foundations of Scientific Inference*

We will be mostly interested in this *important* inferences where premises do not necessitate the conclusion but make it *probable*, which we refer to as *probabilistic inference*.

Here, our focus is on sentences which are *similar* to universally quantified ones. Instead of traditional, logical quantifiers (every, no), we have so called vague (or generalized) quantifiers e.g., most, almost every, many, almost none of, etc.

Most dogs bark.	Goofy is a dog.	$\models$? Goofy barks.
Almost no cat barks.	Tom is a cat.	$\models$? Tom doesn't bark.
Cats rarely bite.	Tom is a cat.	$\models$? Tom doesn't bite.
Dogs usually are smart.	Pluto is a dog.	$\models$? Pluto is friendly.
Many cats sleep.	Tom is a cat.	$\models$? Tom sleeps.
Mice are not lovely.	Micky is an atypical mouse.	$\models$? Micky is lovely.

1.4 Linguistic phenomena involved in probabilistic inferences

Lexical and morpho-syntactic apparatus in natural languages (like English) allow us to express expressions that we've been arguing are probabilistic in nature.

1.4.1 Generalized Quantifiers

Logic-based deductive inference usually involves quantifiers, universal (e.g. every, all, no) and existential (e.g. there is a). (Recap: In classical logical frameworks, the universal and existential quantifiers are inter-definable. A question to think about: Why is that so?

Hint:

- Every vampire is sleeping $\iff$ It is not true that there is a vampire who isn't sleeping
- A vampire is sleeping $\iff$ It is not true that no vampire is sleeping)

Not all quantifiers in natural languages are universal and/or existential. Consider: most, few, many, several, almost none of, etc. These quantifiers are called generalized quantifiers. They usually give rise to information that is vague, uncertain.

- Most doctors are smart.
- A few countries has surplus of Covid-19 vaccines.
- Many countries don't have enough Covid-19 vaccines.
- Almost no vampires are sleeping.

While the above sentences give us certainly rich information about the current state of affairs (aka the world), one cannot be certain when discussing particular instantiations of these sentences. For example:

- Dracula is a vampire. Is he sleeping?

(Probably yes, but one cannot claim it with the full certainty.)

We can construct various quantified propositions in English whose meaning has *probabilistic* flavor. One way of thinking about it is to consider a universally quantified proposition and instead of the universal quantifier take some generalized quantifier; the resultant proposition would not any more state a universal property (e.i., one that applies to every element in a domain), but will rather give rise to somewhat *vague* information.

1.4.2 Adverbs of Frequency

Another source of vagueness are adverbs of frequency, e.g, rarely, usually, regularly, frequently, several times a year, probably etc. They modify a verb phrase and thus quantify over actions/events that take place. They can also cast doubt on categorical information, make it less certain. Consider the following example:

- Almost all vampires are friendly.
- Every vampire is probably friendly.

  Are these two equivalent? Maybe! And if yes, then

probably and the generalized quantifier *almost all* have similar semantic effects.

In the following examples, we combine adverbs of frequency and generalized quantifiers:

- Birds are usually able to fly. (Similar to: Most birds are able to fly.)
- Mammals can rarely fly. (Similar to: Almost no mammal can fly.)

This gives us a basis to treat (at least for purposes of a limited usage) adverbs of frequency in a similar manner as generalized quantifiers. (Adverbs of frequency can be seen as generalized quantifiers over events.)

1.4.3 Graded Adjectives and Comparatives

Natural language allows us to express various kinds of properties, some of which can be characterized in terms of degrees (scale). For example, cold, colder, too cold, etc. That is, we can derive from graded (gradable) properties a way of measuring and comparing objects that these properties can be applied.

```
John is taller than most people.
--------------------------------
$ $? John is tall.

Few people are taller than John.
--------------------------------
$ $? John is tall.

NBA players are taller than most people in U.S.
Few people are NBA players.
Muggsy Bogues is an NBA player.
-------------------------------------------
$ $? Muggsy Bogues is taller than most people in U.S.
```

2 Probability theory, Bayesian Reasoning, Probabilistic programming

Link to Video lecture: `https://youtu.be/XyyPeQ37fhc`

2.1 Motivational Problem (1)

You throw two 6-faced dice.

You observe that the sum is greater than 8

What is the probability that the product is greater than 20?

Solution: `DiceExample.hs`

2.2 Motivational Problem (2)

Assume a big bag, which you know contains a lot of red and blue balls. The contents of the bag is thoroughly mixed. You do not know the proportion of blue and red balls in the bag.

You pick 4 balls at random, putting each ball back in the bag after looking at it. The first three are red, the last one is blue.

What is the probability for your next ball-pick to yield a red ball?

2.3 Elements of Probability theory

2.3.1 Concept: Probability distribution

1. Frequency distribution Consider a coin, with two faces, nominally labeled "heads" and "tails".

   Throw it n times. Consider what you will get.

   - $f(Heads) + f(Tails) = n$

   Consider a die, with 6 faces. Throw it n times. Consider what you will get.

   Let Ω = {1,2,3,4,5,6} We say that Ω is the probability space of x.

   Note:

   - $\sum_{x:\Omega} 1 = 6$
   - The measure of the space is 6.

2. Discrete probability distributions

   One can define the probability of an event P(x) (with x:Ω) as the limit of a frequency distribution f divided by the total number of observations of x, for the number of observations tending to infinity.

   If the coin is "fair", we then expect:

   - P(Heads) = 0.5
   - P(Tails) = 0.5

   For any probability distribution P, over a domain Ω, we expect:

   - $\sum_{x:\Omega} P(x) = 1$

3. Continuous probability distribution Later on, we will mostly turn our attention to set of events Ω which are not discrete. For example, instead of considering whether the coin falls heads or tails, consider *where* it ends up falling, for example as a pair of coordinates.

   - $\Omega = \mathbb{R}^2$

If we'd attempt to use a probability distribution as before, we'd end up with P(x) = 0 for every point. So in this case, each element of Ω is assigned not a probability but a *probability density*.

4. Continuous probability distribution: properties

   If f is the *probability density function* (PDF) of P, the fundamental property becomes:

   - $\int_{x:\Omega} f(x)dx = 1$

   Remark: we'll almost never care about the value of f directly; only its behavior under integrals. That is, the only valid question to ask is the probability of the coin falling "within an area" — not "exactly" at a given point.

2.3.2 Notation

In the scientific literature, the $P(...)$ notation is incredibly overloaded. Let us give a number of overloadings, each in terms of the previous one.

- If C is a subset of Ω, then
  - $P(C) = \sum_{x:C} P(x)$ if Ω is discrete
  - $P(C) = \int_{x:C} PDF(x)$ dx if Ω is continuous
- If $c(x)$ is a condition (Boolean expression),
  - $P(c) = P(\{x \in \Omega \mid c(x)\})$.

  That is, we check the probability of the set of events which makes c true.
- If e is an expression,
  - $P(e == x)$, where x is a distribution which one has to figure out implicitly.

1. Examples
   - $P(Heads \cup Tails)$
   - $P(d > 3 \mid d \in 1, 2, 3, 4, 5, 6)$
   - $P(d) = 1/6$

2.3.3 Dependent and Independent events and variables

- Two events A and B are called *independent* events iff.
  - $P(A \wedge B) = P(A) \cdot P(B)$.
- The probability $P(A \cap B)$ is **not** equal to $P(A) \cdot P(B)$ in general!

2.3.4 Examples

1. Coins
   - $P(Heads \wedge Tails) = 0$ (Indeed, the events are *dependent* on each other)
2. Dice
   - Throw a pair of 6-faced dice d_1, d_2. $P(d_1+d_2 > 9)$ = ?

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

     - 36 (equally probable, aka equiprobable) cases
     - 21 out of 36 satisfy the condition
     - $\longrightarrow P(d_1 + d_2 > 9) = 21/36 = 7/12$

2.3.5 Conditional probability (1)

Definition:

- $P(A \mid B) = P(A \wedge B)/P(B)$
  - if $P(B) > 0$

Example:

$$\begin{aligned} P(Heads \mid Tails) &= P(Heads \wedge Tails)/P(Head) \\ &= 0/0.5 \\ &= 0 \end{aligned}$$

$$\begin{aligned} P(d_1 + d_2 > 6 \mid d_2 = 5) &= P(d_1 + d_2 > 6 \wedge d_2 = 5)/P(d_2 = 5) \\ &= (5/36)/(1/6) \\ &= 5/6 \end{aligned}$$

2.3.6 Conditional probability (2)

Alternatively one can use $P(A \mid B)$ as a primitive notion and define

- $P(A \wedge B) = P(A \mid B) \cdot P(B)$

This equation is useful when $P(A \mid B)$ is known or easy to compute.

1. Example

$$\begin{aligned} P(d_1 + d_2 > 6 \wedge d_2 = 5) &= P(d_1 + d_2 > 6 \mid d_2 = 5) \cdot P(d_2 = 5) && \text{by the above} \\ &= P(d_1 + 5 > 6) \cdot P(d_2 = 5) && \text{by substitution} \\ &= P(d_1 > 1) \cdot P(d_2 = 5) && \text{by subtracting 5} \\ &= (5/6) \cdot (1/6) \\ &= (5/6).(1/6) \\ &= 5/36 \end{aligned}$$

2.3.7 Probability Laws

1. Probability of disjoint events

   A and B are said to be disjoint (as sets or conditions) iff.

   - $A \cap B = \emptyset$
   - $A \wedge B = false$

   If A and B are disjoint, then the probability of the union is the sum of probabilities:

   - If $A \wedge B = false$, then $P(A \vee B) = P(A) + P(B)$
   - If $A \cap B = \emptyset$, then $P(A \cup B) = P(A) + P(B)$

   Remark: Do not confuse "disjoint" and "independent".

2. Probability of negation/complement

   - $P(\neg A) + P(A) = P(\neg A \vee A) = P(true) = 1$

   hence:

   - $P(\neg A) = 1 - P(A)$

   exercise: use sets instead of Boolean expressions.

3. Law of total probability if B_1, B_2 disjoint and $B_1 \vee B_2 = true$:

   - $P(A) = P(A \wedge B_1) + P(A \wedge B_2)$

   Indeed,

$$\begin{aligned} P(A \wedge B_2) + P(A \wedge B_1) &= P((A \wedge B_2) \vee (A \wedge B_1)) && \text{disjoint events} \\ &= P(A \wedge (B_2 \vee B_1)) \\ &= P(A \wedge true) \\ &= P(A) \end{aligned}$$

4. Probability of disjunction

   What if A and B are not disjoint?

   - $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

   Lemma: if $A \subseteq B$ then $P(A) + P(B \setminus A) = P(B)$ Proof:

$$\begin{aligned} P(A) + P(B \setminus A) &= P(A \cup (B \setminus A)) && \text{disjoint events} \\ &= P(B) \end{aligned}$$

   Proof of theorem:

$$\begin{aligned} P(A \cup B) &= P(A \cup (B \setminus A)) \\ &= P(A) + P(B \setminus A)) && \text{disjoint events} \\ &= P(A) + P(B) - P(A \cap B) && \text{Lemma} \end{aligned}$$

5. Summary of Laws

   - $P(\Omega) = 1$
   - $P(\neg A) = 1 - P(A)$
   - $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
   - $P(A \wedge B) = P(A \mid B) \cdot P(B)$
   - if B_1, B_2 complementary, $P(A) = P(A \wedge B_1) + P(A \wedge B_2)$

2.3.8 Random variables with priors (discrete)

How to evaluate $P(A)$, for an expression A depending on a random variable r?

Using the law of total probability:

$$\begin{aligned} P(A) &= \sum_{i:\Omega} P(r = i \wedge A) \\ &= \sum_{i:\Omega} P(A \mid r = i) P(r = i) \end{aligned}$$

We can even write

- $P(A) = \sum_{i:\Omega} P(A[i/r]) P(r = i)$

Writing $A[i/r]$ to mean that we substitute r for i in the expression A. But $A[i/r]$ no longer depends on a random variable. (It is either true or false). So it is less confusing to write *Indicator* instead of P, where $Indicator(c) = 1$ when c is true and 0 when c is false.

- $P(A) = \sum_{i:\Omega} Indicator(A[i/r])P(r = i)$

In such an equation, we can call $P(r = i)$ the prior probability of $r = i$.

1. Example (discrete) In the dice example, every time that we want to evaluate the probability of an event (or condition) A which **depends** on the roll of the dice, we can use the formula:

   - $P(A) = \sum_{i\in[1..6]} \sum_{j\in[1..6]} P(A \mid d_1 = i \wedge d_2 = j)P(d_1 = i \wedge d_2 = j)$

   If the dice are fair and independent, then $P(d_1 = i \wedge d_2 = j) = 1/36$, for any i, j, and we have:

   - $P(A) = \sum_{i\in[1..6]} \sum_{j\in[1..6]} P(A \mid d_1 = i \wedge d_2 = j)/36$

   and even:

   - $P(A) = \sum_{i\in[1..6]} \sum_{j\in[1..6]} Indicator(A[d_1 = i, d_2 = j])/36$

   If the dice were unfair or dependent, we'd change the prior $P(d_1 = i \wedge d_2 = j)$ accordingly.

   Say if A is $d_1 + d_2 > 6$:

   - P(d_1+d_2>6) = Σ(i∈[1..6]) Σ(j∈[1..6]) Indicator(i+j > 6) / 36

2. Random variables with priors (continuous) For continuous variables, we have:

$$\begin{aligned} P(A) &= \int_{x:\Omega} f(r = i \wedge A) f_r(x) dx \\ &= \int_{x:\Omega} Indicator(A[x/r]) f_r(x) dx \end{aligned}$$

   with: f_r the PDF of the distribution of the random variable r.

   Example: probability that the coin falls on the table: Let

   - $A \triangleq (x \in Table)$ (where Table is a subset of $\mathbb{R}^2$ representing the surface of the table.)
   - $f(Coin) \triangleq 1/a$ (using a simple model where I can throw the coin anywhere in the room and a = room areaRoom area.)

$$
\begin{aligned}
P(A) &= \int_{x:\mathbb{R}^2} Indicator(x \in Table)\frac{1}{a}dx \\
&= \frac{1}{a}\int_{x:\mathbb{R}^2} Indicator(x \in Table)dx \\
&= \frac{1}{a}\left(\int_{x\in Table} Indicator(x \in Table)dx + \int_{x\in(\mathbb{R}^2\setminus Table)} Indicator(x \in Table)dx\right) \\
&= \frac{1}{a}\left(\int_{x\in Table} 1dx + \int_{x\in(\mathbb{R}^2\setminus Table)} 0dx\right) \\
&= \frac{1}{a}\left(1\int_{x\in Table} dx + 0\int_{x\in(\mathbb{R}^2\setminus Table)} dx\right) \\
&= \frac{1}{a}\left(\int_{x\in Table} dx\right) \\
&= \frac{1}{a}t \\
&= \frac{t}{a}
\end{aligned}
$$

Any idea of a better model? What would be the effect on the outcome?

2.3.9 Evidence and posteriors

Assume now that we have some **evidence** to account for.

In the case of the dice, we could somehow know that the sum is greater than 8. Then what is the **posterior** probability that the product is less than 20?

- $E \triangleq d_1 + d_2 > 8$
- $A \triangleq d_1 \cdot d_2 > 20$

We need to account for E:

- $P(A \mid E) = P(A \wedge E)/P(E)$
-

$$
\begin{aligned}
&P(d_1 \cdot d_2 < 20 \wedge d_1 + d_2 > 8) = \\
&\qquad \sum_{\mathrm{i}\in[1..6]} \sum_{\mathrm{j}\in[1..6]} Indicator(i + j > 8 \wedge i \cdot j > 20)P(d_1 = i \wedge d_2 = j)
\end{aligned}
$$

2.4 Probabilistic Programs

The above gives an informal recipe to compute probabilities. It works for simple problems, but it's easy to make mistakes when tackling non-trivial problems. We set out to make the process systematic – and so it can also be the basis of complex models. This systematic approach will help with the interpretation of natural language, which is our real goal.

We are really interested in defining spaces of possible situations. We will do so with the help of *probabilistic programs*. Probabilistic programs are procedures whose return value may depend on sampling from a distribution.

Besides, the $P(...)$ notation is a problem on its own, with so much overloading that it's often hard to grasp. Probabilistic programs largely eliminate issues of the $P(...)$ notation.

2.4.1 Probability distributions (1)

The basic characteristic of probabilistic programs is the ability to sample from probability distributions. We will list and discuss some of them.

- $DiscreteUniform(\Omega)$

$$\begin{aligned} P(x) &= 1/\text{measure}(\Omega) && \text{if } x \in \Omega \\ &= 0 && \text{otherwise} \end{aligned}$$

  Suitable if all choices are equally probable — for a finite set of events.

- $Uniform(a, b)$
  - PDF(x) = 1/(b-a) if x ∈ [a,b] = 0 otherwise
  - if all choices are equally probable — if the set of events is continuous and bounded.
- $Bernoulli(p)$
  - $P(0) = 1 - p$
  - $P(1) = p$
  - Two choices, which are not necessarily equally probable.
  - In our example, we can represent the space of Balls by Bernoulli(ρ)

In probabilistic programs, we can sample from a distribution using a special-purpose primitive *sample*. Example:

$$ballBlue = sample\ (Bernoulli\ \rho)$$

(The proportion of balls in the bag is ρ and is unknown)

2.4.2 Probability distributions (2)

- $Normal(\mu, \sigma)$
  - PDF(x) = $\dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
  - Often used to model a random variable which depends itself on many variables in an unknown way

- mean = μ

- $Beta(\alpha, \beta)$ with $\alpha, \beta > 0$
  - $PDF(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$ if $x \in [0,1]$, 0 otherwise
    where $\mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ and Γ is the Gamma function. (just a normalization factor)
  - Useful to model bounded variables, with non-uniform distributions.
  - $Beta(1,1) = Uniform[0,1]$
  - Mean = $\frac{\alpha}{\alpha+\beta}$
  - increasing α "pushes" the distribution towards 1; β towards 0.

2.4.3 Constants

A simple (but very important!) probabilistic program is the one which just returns a constant k. We write it:

return k

2.4.4 Sequencing instructions

If t and u are probabilistic probabilistic programs, then the following is also a probabilistic program:

do $x \leftarrow t; u$

Here, we additionally allow the rest of the program, u, to depend on (use) the variable x. Note that this is the *only construction* that can declare a variable.

ball = **do** *isBlue* ← *sample* (*Bernoulli* ρ);
return (**if** *isBlue* **then** *Blue* **else** *Red*)

2.4.5 Observations

To represent evidence, we introduce the program *observe* (φ), where φ is a Boolean-valued expression. If φ is true, then *observe* (φ) has no effect. If φ is false, then *observe* (φ) then the program is aborted; in essence the samples made above in the program are discarded. We will make formally precise later.

In our running example, a program sampling a blue ball can be written as:

blueBall = **do**
$x \leftarrow$ *ball*

$observe\ (x \equiv Blue)$
$return\ x$

A program sampling two balls, and at least one blue, is:

$twoBallsAtLeastOneBlue =$ **do**
$x \leftarrow ball$
$y \leftarrow ball$
$observe\ (x \equiv Blue \vee y \equiv Blue)$
$return\ (x, y)$

2.4.6 Expected truth value (aka "Probability")

We can now conveniently phrase our problems in this framework:
If we let $die = sample\ (DiscreteUniform\ [1\,..\,6])$
The program representing situations where the sum of dice is > 8 is :

$twoDieAbove8 =$ **do**
$d_1 \leftarrow die$
$d_2 \leftarrow die$
$observe\ (d_1 + d_2 > 8)$
$return\ (d_1, d_2)$

What want to do now is to sample a pair of dice using the above procedure, then evaluate the probability that the product is greater than 20.

Given a random pair (x, y) sampled by *twoDieAbove6*, we'd be interested in the truth value of the proposition

- $\varphi = x \times y > 20$

But φ depends on which pair (x, y) we choose. So the *probability* of φ is given by the expected value of the indicator function $Indicator(\varphi)$.
So we could define the probability of φ as:
$\mathbb{E}_{[(x,y) \in twoDieAbove8]}(Indicator(x \times y > 20))$
There is however a convenient way to represent the above expression in terms of a probabilistic program directly:

$problem1 =$ **do**
$(x, y) \leftarrow twoDieAbove8$
$return\ (indicator <\$> (x * y > 20))$

This way, to evaluate probabilities, the only thing that we need is to take the expected value of probabilistic programs. (We will see later how to do this.)

1. Remark (skip)

do $x \leftarrow a$
$observe\ (b\ x)$
$return\ (c\ x)$

is not the same as

```
do x ← a
   return (b x ⟶ c x)
```

In the first instance, if $b(x)$ is false so x is not counted. In the 2nd program if $b(x)$ is false it is counted as satisfying the condition.

2.4.7 Example: drug test (Wikipedia)

> Suppose that a test for using a particular drug is 99% sensitive and 99% specific. That is, the test will produce 99% true positive results for drug users and 99% true negative results for non-drug users. Suppose that 0.5% of people are users of the drug. What is the probability that a randomly selected individual with a positive test is a drug user?

This can be modeled by the following probabilistic program:

```
exampleDrug = do
    -- prior
  isUser ← sample (Bernoulli (0.5 * percent))
    -- evidence
  testedPositive ← if_ isUser $ λcase
    True → sample (Bernoulli (99 * percent))
    False → sample (Bernoulli (1 * percent))
  observe testedPositive
    -- posterior
  return isUser
```

(Complete program file here: `DrugTest.hs`)

2.4.8 Answer To Introductory Problem

link back to the introductory problem

One might think that a simple answer is $\frac{3}{4}$. But is this correct? Let's try to use the concepts developed so far and write a probabilistic program modeling the problem:

```
exampleBalls = do
    -- a priori distribution of the proportion of blue balls.
  ρ ← sample (Uniform 0 1)    -- ρ ← sample (Beta 0.5 0.5) – alternative
    -- sample a ball in the bag:
  let ball = do
    x ← sample (Bernoulli ρ)
    return (boolToColor<$>x)
    -- sample a red ball:
  let redBall = do
    b ← ball    -- take a ball
```

```
    observe (testEq b Red)    -- if it is not red, forget this situation.
    -- sample a blue ball:
  let blueBall = do
    b ← ball
    observe (testEq b Blue)
  redBall
  redBall
  redBall
  blueBall
  x ← ball
  return x
  where boolToColor : : Bool → Color
    boolToColor = λcase
      True → Blue
      False → Red
```

What do you think is the expected result of this program? This is the topic of the rest of the lecture.

2.4.9 Meaning of probabilistic programs

Intuitively, probabilistic programs define distributions. However, defining distributions directly poses a number of technical problems. So instead we define the related notion of integrator.

We define the *integrator* of $f(z)$ over a probabilistic program t, $(\oint_{z \in t} f(z))$ as a generalization of the integration/summation of $f(x)$ for the possible return values z returned by t.

We define:

$$\oint_{z \in return\ x} f(z) = f(x)$$

$$\oint_{z \in (do\ x \leftarrow t; u(x))} f(z) = \oint_{x \in t} \oint_{z \in u(x)} f(z)$$

$$\oint_{z \in sample(c)} f(z) = \int_{x \in \mathbb{R}} \mathrm{PDF}_c(x) \cdot f(x) dx \quad \text{sampling in a continuous distribution}$$

$$\oint_{z \in sample(d)} f(z) = \sum P_d(x) \cdot f(x) \quad \text{sampling in a discrete distribution}$$

$$\oint_{z \in observe(\varphi)} f(z) = Indicator(\varphi) \cdot f(\diamond)$$

(Note that the "observe" program does not return any result, so the integrand $f(z)$ cannot in fact depend on z, the result of the program.)

Don't worry if you don't get all details at this stage. The main point is that we can define the meaning of probabilistic programs in a precise, mathematical manner. (Without referring to how probabilistic programs are run on an actual machine.)

1. Measure

   Probabilistic programs do **not**, in general, define distributions. That is, the total *measure* of a program is not guaranteed to be 1. For example, the program *observe false* has a measure of 0.

   We define the measure of a program t as follows:

   $\mathsf{measure}(t) = \sum_{z \in t} 1$

   Thus the measure "counts" every element with the same unit weight.

2. Lemma: integrators are linear operators

   [In the vector space of real-valued functions]

   Lemma:

   - $\int_{x \in t} (k \times f(x)) = k \times \int_{x \in t} f(x)$
   - $\int_{x \in t} (f(x) + g(x)) = \int_{x \in t} f(x) + \int_{x \in t} g(y)$

   Proof: By induction on t, relying on the linearity of $\sum$ and $\int$.

3. Lemma: Properties of measures

$$\begin{aligned} \mathsf{measure}(\mathit{sample}\ d) &= 1 \\ \mathsf{measure}(\mathit{observe}\ \varphi) &= \mathit{Indicator}[\![\varphi]\!] \\ \mathsf{measure}(\mathit{do}\ x \leftarrow t; u) &= \int_{x \in t} \mathsf{measure}(u) \end{aligned}$$

   - Proposition: For probabilistic program t, $\mathsf{measure}(t) \leq 1$.

4. Expected value

   The expected value of $f(z)$ over a value z sampled by a probabilistic program t is defined as follows:

   $$\frac{\int_{z \in t} f(z)}{\mathsf{measure}(t)}$$

   It is also very useful to define the expected value of a probabilistic program itself, as simply the expected value of its returned values:

   $\mathbb{E}(t) = \mathbb{E}_{z \in t}[z]$

   (This make sense only when t returns a numerical value.)

5. Expected truth value (aka "probability")

   If a program t returns the type *Bool* (either the constant *True* or *False*), we can define the probability of t (to return *True*) as:

   $\mathbb{P}(t) = \mathbb{E}[z \leftarrow t; return(Indicator(z))]$

   (So we simply convert the Boolean value to 0 or 1, and then take the expected value.)

6. Example: Drug test Given *exampleDrug* defined as above (`DrugTest.hs`)

   - Compute: $\mathbb{P}(exampleDrug)$
   - Answer: $\frac{\sum_{z \in t} indicator(z)}{\mathsf{measure}(exampleDrug)}$

7. Exercise: compute the above answer using the definitions.

   Solution = $\frac{ok}{total}$

$$
\begin{aligned}
ok &= \mathsf{measure}(t) \\
&= \sum_{isUser : Bool} Bernoulli(0.005)(isUser) \cdot \sum_{testPositive : Bool} Bernoulli(if\ isUser\ then\ 0.99\ else\ 0.01)(testPositive) \cdot Indicator \\
&= \sum_{isUser : Bool} Bernoulli(0.005)(isUser) \cdot Bernoulli(if\ isUser\ then\ 0.99\ else\ 0.01)(true) \\
&= \sum_{isUser : Bool} Bernoulli(0.005)(isUser) \cdot if\ isUser\ then\ 0.99\ else\ 0.01 \cdot \\
&= Bernoulli(0.005)(false)(if\ \ false\ then\ 0.99\ else\ 0.01) + Bernoulli(0.005)(true)(if\ true\ then\ 0.99\ else\ 0.01) \\
&= 0.995 \times 0.01 + 0.005 \times 0.99 \\
&= 0.0149
\end{aligned}
$$

Compute the numerator:

$$
\begin{aligned}
total &= \sum_{isUser : Bool} Bernoulli(0.005)(isUser) \cdot \sum_{testPositive : Bool} Bernoulli(if isUser then 0.99 else 0.01)(testPositive) \cdot Indicator(\\
&= Bernoulli(0.005)(true) \cdot Bernoulli(if true then 0.99 else 0.01)(true) \\
&= 0.005 \times 0.99 \\
&= 0.00495
\end{aligned}
$$

So the ratio is: 0.332214765101

2.4.10 Exercise: Evaluating the answer introductory problem

Solution.
$\frac{ok}{total}$ with:

$$
\begin{aligned}
ok = \int_{\rho:[0..1]} d\rho \\
&\sum_{b1:[0,1]} b(\rho, b1) \cdot \\
&\sum_{b2:[0,1]} b(\rho, b2) \cdot \\
&\sum_{b3:[0,1]} b(\rho, b3) \cdot \\
&\sum_{b4:[0,1]} b(\rho, b4) \cdot \\
&\sum_{b5:[0,1]} b(\rho, b5) \cdot \\
&(b1 \cdot b2 \cdot b3 \cdot (1 - b4) \cdot b5)
\end{aligned}
$$

$$
\begin{aligned}
total = \int_{\rho:[0..1]} d\rho \\
&\sum_{b1:[0,1]} b(\rho, b1) \cdot \\
&\sum_{b2:[0,1]} b(\rho, b2) \cdot \\
&\sum_{b3:[0,1]} b(\rho, b3) \cdot \\
&\sum_{b4:[0,1]} b(\rho, b4) \cdot \\
&(b1 \cdot b2 \cdot b3 \cdot (1 - b4))
\end{aligned}
$$

(Computing the integrals is daunting! But can your algebra system do it?)
Reminder: Where:

- $b(\rho, 0) = \rho$
- $b(\rho, 1) = 1 - \rho$

2.4.11 Final note on Beta distribution.

If we observe n reds and m blues, the posterior distribution for the parameter ρ is $Beta(n + 1/2, m + 1/2)$.

In particular, the expected value of this proportion is $\frac{n+0.5}{n+m+1}$

In our example, we do not expect to a $3/4$ prediction for the ratio of red ball, but rather but $3.5/5$. (Which is exactly what the program predicts!)

2.4.12 Exercise: the children problem

Model the following problem:

A friend of yours has exactly two children. One of them is a boy. What is the probability that the other one is a boy?

Solution: `Pair.hs`

2.4.13 Exercise: betting on games

- Consider the game "Sloubi".
- Each player p of Sloubi is assigned a rating ρ_p. The rating is intrinsic to each player, and never changes.
- There is an element of randomness in Sloubi. In any match, p will win over q if $\rho_p > \rho_q + m$, with m taken in $Normal\ (0, 100)$. (Even worse players will win, sometimes.)
- Alice wins over Bob, Bob wins over Charles and David. What is Alice's probability to win over David in their next game?

1. Solution

   `Sloubi.hs`

3 Compositional translation of inference problems into probabilistic programs

Link to Video lecture: `https://youtu.be/XJugaNpNi_0`

3.1 Montagovian semantics

- Note: a comprehensive course was given in the first week of ESSLLI 2021 "Introduction to natural language formal semantics", by Ph. de Groote and Y. Winter.

As a quick reminder, we can associate types with syntactic categories, in the following manner:

type $CN = Ind \to Prop$
type $VP = Ind \to Prop$
type $NP = VP \to Prop$
type $Quant = CN \to NP$
type $Ind = ...$

But what are individuals? It is mysterious!

In fact, Montagovian semantics normally consider Individuals to be *abstract*. This means that nothing needs to be known about them to be able to interpret phrases. However, if one needs to give specific semantics to lexical items (perhaps in specific domains), we need to get more concrete.

In fact we keep (nearly) all Montagovian semantics as such, and make certain things concrete.

3.2 Interpreting inference problems: recipe.

Remember the classic syllogism:

- all men are mortal
- socrates is a man
- socrates is mortal?

It can be interpreted as probabilistic program this way:

```
exampleSocrates0 = do
  man ← samplePredicate        -- declare predicate
  mortal ← samplePredicate     -- declare predicate
  observe (every man mortal)   -- premiss, interpreted using Montegovian semantics
  socrates ← sampleInd         -- declare individual ("for some random ...")
  observe (man socrates)       -- premiss, interpreted using Montegovian semantics
  return (mortal socrates)     -- conclusion, interpreted using Montegovian semantics
```

That is:

- every time we have a new lexical item, we sample it at random
- every time we have a premiss, we observe it to be true
  - this means that samples which do not satisfy the premiss will be rejected
- at the end, we return the truth value of the conclusion
  - so the probability of the program corresponds to the probability of entailment.

On this example, this means that we quantify *man* over all CNs; so the program does not have any *a-priori* notion of what "man" means — we sample over the whole space of CNs. The distribution for "man" gets refined by evidence (in this case "∀(x:man) mortal(x)", "man(socrates)"). The way, the semantics for programs that we gave mean that all worlds where we can find non mortal men will be filtered out.

3.3 Interpretation of semantic categories

We still need to chose a definition for *samplePredicate*, *sampleInd*.

One would expect the above inference regarding Socrates to hold in every possible world. Consequently, we'd like $\mathbb{P}[exampleSocrates0] = 1$ So, the definitions of CN,VP,Ind, etc. must be well-chosen so that the above formula evaluates to 1.

The goal is to

- interpret each syntactic category as a probabilistic program
- interpret each syntactic operator as a function from/to the appropriate spaces.
- so that we get meaningful inferences

3.3.1 Propositions

We'll simply interpret propositions as Boolean-valued expressions.

type *Prop* = *Probabilistic Bool*

3.3.2 Individuals

Fortunately we now have a way to interpret individuals as elements in a space.

Examples:

- multi-variate normal distribution of dimension k
  - covariance matrix (?)
- uniform distribution in a box $[0..1]^k$

  sampleInd = *sampleVectorOf* (*Gaussian* 0 1)

- Discussion: what would *you* choose? Why?

This idea is directly inspired from machine learning: individual (situations) can be represented by a vector.

This is indeed used for:

- Words
- Sentences
- Images

3.3.3 Reminder: set cardinalities

If $card(A) = n$, then $card(A \to Bool) = 2^{card(A)}$. This is because if a set has n elements, then it has 2^n possible subsets. So, there are "exponentially many" more predicates over a set than there are elements in the set.

A related fact is that $\mathbb{N}$ is countable, but the set of predicates over natural numbers $\mathbb{N} \to Bool$ is uncountable.

There is an obvious way to integrate over $[0, 1]$, but how to integrate over $[0, 1] \to Bool$? How to take "the average" over all possible predicates?

3.3.4 Space of predicates

We're deliberately going to restrict the set of possible predicates to make our endeavor possible. Hopefully, it's enough to limit oneself to a small enough (sampleable) subset and still have a useful model.

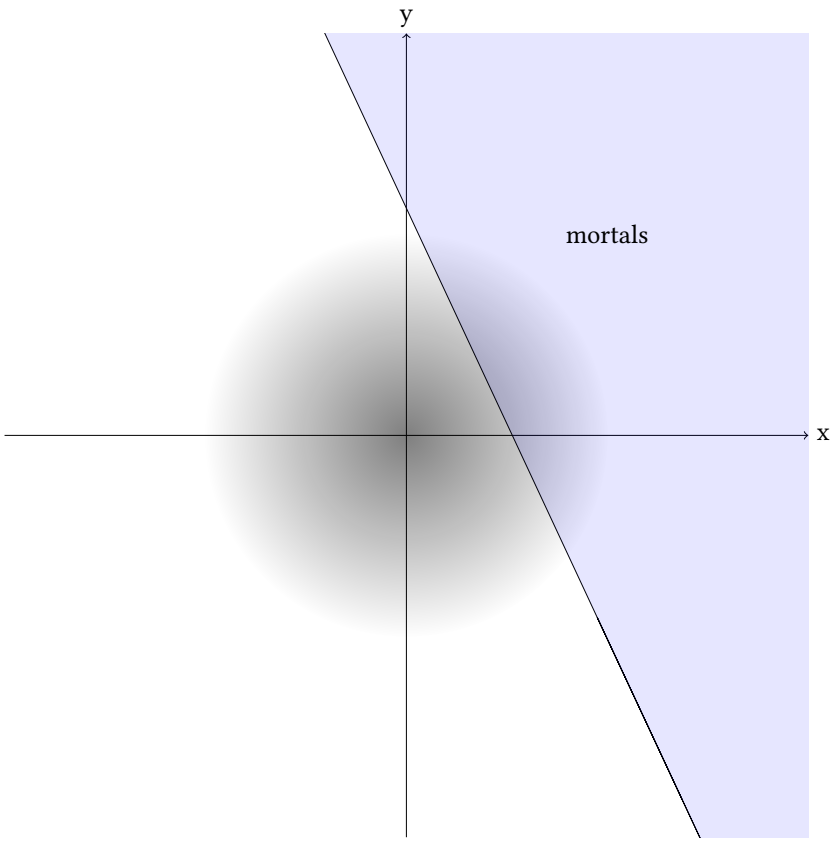
If words can be represented by a vector, then so can predicates (hopefully).

(NOTE: other ideas would be to sample from a set of programs which implement predicates.)

3.3.5 Idea 1

If an individual is represented by a vector x and a vector p represents a predicate, then x is said to satisfy the predicate if $p \bullet x > 0$. (Ie, both vector are oriented in the same direction in the underlying euclidean space.)

$$
\begin{aligned}
&predicateSimple = \textbf{do} \\
&\quad v \leftarrow sampleNormedVector \\
&\quad b \leftarrow sample\ (Gaussian\ 0\ 1) \\
&\quad return\ (\lambda x \to (b + x \cdot v) > 0)
\end{aligned}
$$



Here, the blue area represents the subspace where individuals satisfy the predicate.

3.3.6 Idea 2: Boxes

If an individual is represented by a vector x and a pair of vectors p , q represent a predicate, then x is said to satisfy the predicate if x is in the box delimited by the corners p and q.

$$
\begin{aligned}
&\mathit{predicate} = \textbf{do} \\
&\quad p \leftarrow \mathit{sampleVectorOf}\,(\mathit{Gaussian}\ 0\ 1) \\
&\quad q \leftarrow \mathit{sampleVectorOf}\,(\mathit{Gaussian}\ 0\ 1) \\
&\quad \mathit{return}\ (\lambda x.\ \forall\ i.p_i < x_i < q_i)
\end{aligned}
$$

3.3.7 Idea 3: Your Idea!

Discussion point

3.3.8 Common nouns

Common nouns are interpreted as predicates:

$[\![CN]\!] = Pred$

Consequently, any given common noun *cn* is a predicate. We can also interpret the common noun *cn* as the underlying sub-distribution of individuals which is filtered by satisfying the predicate associated with *cn*, like so.

$$
\begin{array}{l}
sampleSome\ cn = \textbf{do} \\
\quad x \leftarrow sampleInd \\
\quad observe\ (cn\ x) \\
\quad return\ x
\end{array}
$$

3.4 Interpretation of semantic operators

3.4.1 Generalized quantifiers

We can defined generalized quantifiers by appealing to the measure of probabilistic programs:

- $atLeast\ \theta\ a\ (\lambda x.\varphi) = measure\ (\textbf{do}\ x \leftarrow a; observe\ (\varphi)) > \theta * measure\ (a)$
- $atMost\ \theta\ a\ (\lambda x.\varphi) = measure\ (\textbf{do}\ x \leftarrow a; observe\ (\varphi)) < \theta * measure\ (a)$

Indeed, the statement $observe\ (\varphi)$ will discard certain samples of x, and affect the measure in proportion to the probability of $\varphi\ (x)$ to hold.

Note that this requires to *evaluate the measure of a program inside a probabilistic program itself.*

However so far we have seen only how to evaluate their probabilities. No sweat, we can use an alternative definition, as follows:

Equivalently:

$$atLeast\ \theta\ cn\ vp = probability\ (\textbf{do}\ x \leftarrow sampleSome\ cn; return\ (vp\ x)) > \theta$$

In turn we can define all sorts of generalized quantifiers:

- $[\![Most\ cn\ vp]\!] = atLeast\ \theta\ [\![\ cn\ ]\!]\ (\lambda x.\ [\![\ vp\ ]\!]\ (x))$
- $[\![Few\ cn\ vp]\!] = atMost\ (1-\theta)\ [\![\ cn\ ]\!]\ (\lambda x.\ [\![\ vp\ ]\!]\ (x))$

1. Example:
    - most men are mortal
    - socrates is a man
    - $\models$? socrates is mortal

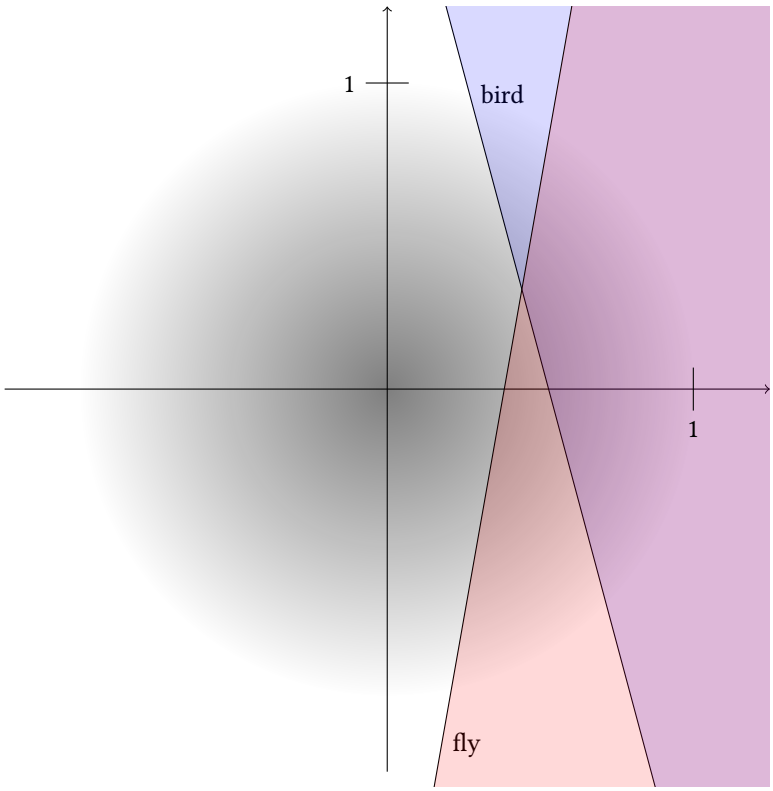
$$
\begin{array}{l}
exampleSocrates = \textbf{do} \\
\quad man \leftarrow samplePredicate \\
\quad mortal \leftarrow samplePredicate \\
\quad observe\ (most\ man\ mortal)
\end{array}
$$

```
socrates ← sampleInd
observe (man socrates)
return (mortal socrates)
```

2. Example:

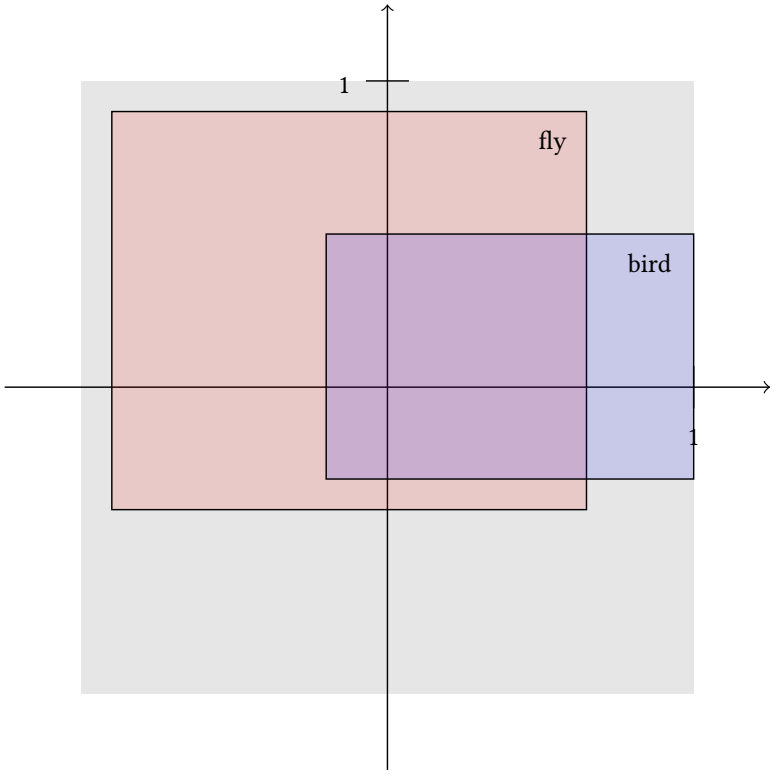
   - Few animals fly.
   - Most birds fly.
   - Every bird is an animal.
   - $\models$? most animals are not birds

```
exampleBirds = do
  fly ← samplePredicate
  bird ← samplePredicate
  animal ← samplePredicate
  observe (most bird fly)
  observe (few animal fly)
  observe (every bird animal)
  return (most animal (λx → ¬<$>(bird x)))
```



Here, the blue-shaded area represents the subspace where individuals satisfy the **bird** predicate. So the greyed out part shaded in blue represents birds. The red-shaded area represents the subspace where individuals satisfy the **fly** predicate. The intersection (hence shaded in purple), represent the subspace of individuals which satisfy both predicates.

Same example, but using boxes for predicates:



3. Choice of θ

   The above depends on a threshold θ which constitutes the proportion from which most/few/etc. begin to hold.

   One can choose θ by studying native speakers. However, one should expect that you won't get a single value of θ which will fit all situations, but rather you'll observe a distribution for θ. Probabilistic programs are ideally suited to deal with this.

```
exampleSocrates2 = do
  θ ← sample (Beta 5 2)    -- for example
  man ← samplePredicate
  mortal ← samplePredicate
  observe (atLeast θ man mortal)
```

$socrates \leftarrow sampleInd$
$observe\ (man\ socrates)$
$return\ (mortal\ socrates)$

Below we'll leave θ abstract.

3.4.2 Universal Quantifiers

We define $forAll\ a\ \varphi$ as stochastic certainty of $\varphi\ (x)$ for elements given by the probabilistic program a, namely:

$$forAll\ a\ \varphi = probability\ (\textbf{do}\ x \leftarrow a; return\ (\varphi\ x)) \equiv 1$$

Given the above, we can interpret natural language phrases such as "every man is mortal", as follows:

- $[\![Everycnvp]\!] = forAll[\![cn]\!][\![vp]\!]$

1. Pitfall (SKIP)

   Assume

   - $t = sample(Uniform[-1..1])$
   - $\varphi = (x \neq 0)$

   We have:

   - $\mathsf{measure}(t) = 2$
   - $\mathsf{measure}(x \leftarrow t; observe(\varphi)) = 2$

   Indeed, we filtered out a single point — its measure is 0

   And according to the above definition:

   - $[\![\forall(x\ :\ A).\varphi]\!] = true$

   (So this operator really means "for stochastically all" in probabilistic logic)

   (a) Dealing with this pitfall

   - attempt to have a precise measure that counts single elements
     - not computable, because HOL is undecidable
   - use "soft transitions"
     - still does not make $\forall x\ :\ A.\varphi$ coincide with the usual definition (but can help with the approximation algorithms in many cases.)
   - do not use problematic domains
     - unless otherwise note, this is what we will do.

3.4.3 Existential Quantifiers

One can define existential quantifiers by dualizing universals in either version.

$$exist\ a\ \varphi = probability\ a\ \varphi > 0$$

3.4.4 Comparatives

- Mary is tall
- John is tall
- "Mary is taller than John"?

We can support graded predicates and comparatives. We do so by generalizing predicates.

We define *Grade* to be function from individuals to reals.

$$\textbf{type}\ Grade = Ind \to Probabilistic\ R$$

If the function evaluates to a positive value for individual x, then x is considered to satisfy the non-scalar retraction of the predicate.

$$is :: Grade \to Ind \to Prop$$
$$is\ g\ x = g\ x > 0$$

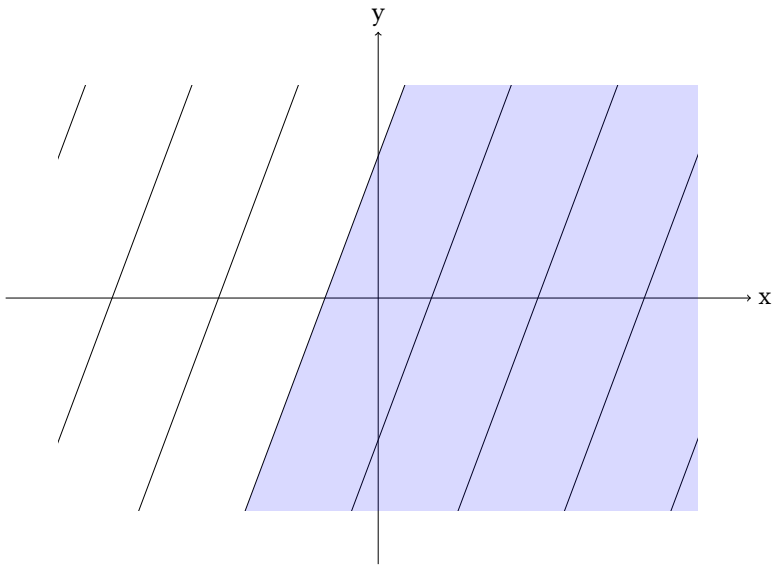
Then one can also compare individuals with respect to any scalar predicate:

more :: Grade ⟶ Ind ⟶ Ind ⟶ Prop more g x y = g x > g y Eg. to test *more tall mary john*, we check if the 'tallness' of mary is greater than that of john.

1. Idea 1

   The expression $b + d \cdot x$ can be interpreted as a degree to which the individual x satisfies the property characterized by (b, d).

$$\begin{array}{l} sampleGrade = \textbf{do} \\ \quad v \leftarrow sampleNormedVector \quad \text{-- reference vector for the grade} \\ \quad b \leftarrow sample\ (Gaussian\ 0\ 1) \quad \text{-- reference bias} \\ \quad return\ (\lambda x \to (b + x \cdot v)) \end{array}$$



2. Idea 2 The degree to which an individual x satisfies a property characterized by a box centered at c and of dimensions d is given by $s(x)$.

   The characteristic vectors c and d are sampled from the vector space. Thus we get:

   $$
   \begin{array}{l}
   \mathit{grade} :: P\ \mathit{Grade} \\
   \mathit{grade} = \textbf{do} \\
   \quad c \leftarrow \mathit{sample}\ (\mathit{Gaussian}\ 0\ 1) \\
   \quad d \leftarrow \mathit{sample}\ (\mathit{Gaussian}\ 0\ 1) \\
   \quad \textbf{let}\ s\ x = 1 - \mathit{max}\ [\ \mathit{abs}\ (x_i - c_i)\ /\ d_i \mid i \leftarrow [1\,..\,n]\,] \\
   \quad \mathit{return}\ s
   \end{array}
   $$

   This definition entails that the subspace corresponding to a predicate coincides with the space where its degree of satisfaction is positive.

   Remarks:

   - $s(x) = 1$ iff. x is at the center of the box.
   - $s(x) > 0$ iff. x is inside the box.

3. Example

   That is, if we observe that "John is taller than Mary", we will infer that "John is tall" is slightly more probable than "John is not tall".

   $$
   \begin{array}{l}
   \mathit{exampleTall} = \textbf{do} \\
   \quad \mathit{tall} \leftarrow \mathit{sampleGrade} \\
   \quad \mathit{john} \leftarrow \mathit{sampleInd} \\
   \quad \mathit{mary} \leftarrow \mathit{sampleInd}
   \end{array}
   $$

```
observe (more tall john mary)
return (is tall john)
```

3.5 Subsective Graded Adjectives

3.5.1 Reminder: intersective vs subsective adjectives

We could interpret "socrates is a large man", as $man\ socrates \wedge large\ socrates$. This is what we call an intersective interpretation.

However, this can pose problems. Consider the following set of statements:

- Dumbo is not a large elephant
- Mickey is a large mouse
- Dumbo is larger than Mickey

If Dumbo is not large, but Mickey is large, so how could Dumbo be larger than Mickey? The intuitive answer is that when we say "Mickey is a large mouse", we mean that Mickey is large **for** a mouse. But he can still be small compared to Dumbo (who is not a mouse)!

3.5.2 Probabilistic interpretation of subsective adjectives

As a way forward, we are going to give an interpretation of "mickey is large for a mouse", by looking for the average size of a mouse, and check that the size of mickey is greater than that.

```
subsectiveIs g cn x = g x > averageFor g cn
```

and:

```
averageFor g cn = expectedValue<$>mcmc 1000 sampleGForCn
  where sampleGForCn = do
    y ← sampleSome cn
    return (g y)
```

And we can now interpret meaningfully inference problems such as:

- Dumbo is not a large elephant
- Mickey is a large mouse
- Most elephant is larger than most mice
- $\models$? Dumbo is larger than Mickey

We do so as follows:

```
exampleDumbo = do
  elephant ← samplePredicate
  mouse ← samplePredicate
  dumbo ← sampleSome elephant
  mickey ← sampleSome mouse
  large ← sampleGrade
  observe (¬<$>(subsectiveIs large elephant dumbo))
  observe (subsectiveIs large mouse mickey)
  observe (most mouse (λx → most elephant (λy → more large y x)))
  return (more large dumbo mickey)
```

4 Computations: MCMC, Gradients

Link to video lecture: `https://youtu.be/ED4i5cF6nPk` So far, we can:

- evaluate natural language problems to probabilistic programs
- evaluate the expected value of such programs as mathematical formulas

In this chapter we'll learn how to evaluate such formulas to numbers.

4.1 Exact evaluation

Represent the meaning of a probabilistic program as a list of pairs: (*outcome*, *probability*).

Then:

```
⟦
return ⟧k     =   [(k,1)]⟦
x ← t; ⟧u     =   [(y,p×q) \ensuremath{(\Varid{x},\Varid{p})\mathbin∈{}\mathbin⟦{
]⟦
observe φ⟧    = ◇  [(,Indicatorφ())]
```

This yields an exhaustive list of all possible outcome (a new finite distribution).

See haskell implementation here: `Exact.hs`

But, this cannot work for continuous distributions!

4.1.1 Example: drug test

`DrugTest.hs`

4.2 Continuous case

- The formulas we got in the previous chapters involve integrals which are typically not easy to compute (when involving non-trivial spaces).

Fortunately there are ways to approximate probabilities directly without resorting to symbolic integration.

4.3 Markov Chain Monte Carlo

4.3.1 Monte Carlo methods:

Assume a probabilistic program t returning a value in [0,1]. To evaluate $\mathbb{E}(t)$:

- `n := 0; q := 0`
- repeat:
  - take a random output x of t by sampling from t
    - !! If the output is discarded (according to observe), then try again
  - `n := n + 1`
  - `q := q + x`

After sufficiently many trials:

- $\mathbb{E}(t) \approx \frac{q}{n}$

4.3.2 Markov Chain

- Informally: "a random walk"
- assume set of states S, and a starting state s_i.
- for each pair of states (s, t), assume a probability $P(s, t)$ to transition from s to t.
- at each step, transition from a state to another according to the given probabilities.
- an interesting question: after an infinite number of steps, what is the probability to end at a given state s_f?
  - if T is the transition matrix and s_i the initial state, $T^\infty s_i$

4.3.3 MCMC

Sampling in complicated probabilistic program t is not so easy. Typically we have a space with a number of dimensions (the Cartesian product of a number of distributions), and a complicated filtering function (we can have an *observe* statement which depends on a complicated condition.)

One way to improve on this method is to **define** a Markov Chain where:

- each state is an element of the space sampled by t.
  - for example if we sample two variable x and y, (x, y) will be one of these possible states.
  - The probabilistic program gives a probability for the final outcome of every state. Say $P_t(x, y)$.

- We can setup the random walk between states so that it's probable to walk from a state (x, y) to a state (x', y'), if $P_t(x', y')$ is larger $P_t(x, y)$
- This way, after many steps, we are likely to be on a probable state.
- Additionally, once we have found a possible state (one which passes all `observe` statements) we can find a new state by taking one step of the random walk. There is no need to "start from scratch" as in the naive Monte Carlo approach.

We use this kind of random walk to sample in t, and apply the Monte Carlo method as usual to evaluate the proportion.

Potential issues:

- Defining the walk is not too easy when there are many variables (also, with if statements (and *observe*), the existence of variables depend on the value of others.)
- You never find a valid $x \in t$ to start with
- The space is divided in regions which are not connected, or a walk from one to the other is highly improbable.

4.3.4 The Structure of the probabilistic program influences performance

In any program portion $x \leftarrow t$; *observe* φ (x) a potential pitfall is to chose t too wide, (eg. x could be a tuple of many independent variables), followed by a very restrictive φ. In such situation the Monte Carlo algorithm will spend a lot of time sampling elements of t only to discard them. It is better to restrict t to one of its subspaces u so that φ becomes more easy to satisfy on u.

4.3.5 Inner evaluation of proportions

When using quantifiers, we evaluate more proportions/measures, and an inner instance of the MCMC algorithm must be employed. This can be very slow! A potential way out: when we use boxes some integrals can be computed symbolically and we save much resources.

4.4 More methods!

Any mixture of the above is thinkable. (Sample and do gradient descent to determine most probable value of the samples, etc.)

4.5 Returning to our motivating example

`Balls.hs`

4.5.1 Choice of prior (again)

The (Uniform 0 1) prior is biasing the result towards 1/2. In order not to bias the result, one should use the Jeffrey's prior, which in this case is $Beta(0.5, 0.5)$.

The Jeffrey's prior is given by the square root (the determinant of) the Fisher information (matrix) I. (I = variance of the derivative of log of density.)

4.6 More Probabilistic programming packages

- STAN: `https://mc-stan.org/`
- WebPPL: `http://dippl.org/`
- And many, many, many papers about industrial strength and idealized probabilistic programming languages.

5 Test suite for Probabilistic Inference, Outlook

Link to the video lecture: `https://youtu.be/E_itLhgh7Yk`

5.1 Building a corpus of probabilistic inference

Data driven approaches need corpora to learn. But, a corpus is also important for testing systems to establish their strength, weakness, robustness, etc. For natural language inference, there are several corpora, including FraCaS and SICK. However, apart from a corpus designed in CLASP, no corpus exists for probabilistic inference problems.

5.1.1 Generalized quantifiers

Quantification with most, few, many, several, etc. can give rise to non-logical inference, which we called above non-demonstrative inference—ones whose premises do not necessitate the conclusion. While in building a testsuite for probabilistic inference they are irreplaceable, we should limit ourselves not to overuse them as their complex semantic nature does not make easily available various meanings, compared to cases where only classical, universal and/or existential quantification is used. Indeed, consider:

- Every man has a bicycle. (Not too hard to comprehend its meaning albeit ambiguous)
- Every man can ride every bicycle. (Not too hard to understand)
- Most men can ride most bicycles. (What does that mean exactly?)
- Many men love cars but not always can afford them. (Even more obscure than the previous one.)

5.1.2 Bare plurals, Generics, and Indefinite Noun phrases

One has to be careful when dealing with bare plurals as they have (at least) two kinds of meanings: They serve as generalized quantifiers (like most) and they also may carry generic meaning, as it is in the following exammples:

- Ducks lay eggs. (Generic because it's not equivalent to Most ducks lay eggs or all ducks lay eggs)
- Small ducks do not trust humans. (Generalize quantifier usage ~ most small ducks don't trust humans)
- Honeybees produce beeswax. (Generic because, for example, queens do not produce any beeswax)

While indefinite noun phrases in Montague Grammar are translated with the help of existential quantifier, they also show some traits of generic readings:

- A lion has a mane. (Generic: only adult male lions do)
- An adult male lion has a mane. (~ All typical adult male lions have manes)
- A lion saw a zebra. (~Some lion saw some zebra)

5.1.3 Adverbs of frequency

Adverbs of frequency, e.g., seldom, frequently, rarely, couple of times per day, etc. are context dependent:

John frequently goes to Paris.	(e.g. once in a month?)
John frequently goes French restaurants for lunch.	(e.g. twice a week?)
John frequently gets distracted during a lecture.	(e.g. almost always when the teacher starts speaking about r
John frequently sips water during a workout session.	(e.g. three or four times per 45 minutes?)

Adverbs of frequency interact with quantifiers. It's important to ensure their swift interaction in order to make a meaningful test suite example.

5.1.4 Adjectives and Comparatives derived from them

Consider:

```
Few people are basketball players.
Basketball players are taller than most non basketball players.
John is a basketball player.
)===>
John is taller than many people.
```

Consider another example:

```
John is taller than Mary.
Mary is taller than Bob.
)===>
John is taller than Bob.
```

Another question: Is John tall?

5.2 Evaluation criteria of a probabilistic inference system against the corpus

How to evaluate an inference, on what scale: `Yes No`, `Yes NA No`, `Yes Kind of Yes NA Kind of No No`, etc. Our scale: $[0, 1]$ Valid inference has a probability $0.5 + x$ Invalid inference has a probability $0.5 - y$.

```
Most vampires are sleepy.
Dracula is a vampire.
)===>
Dracula is sleepy.     (Probable with d1)

Almost all vampires are sleepy.
Dracula is a vampire.
)===>
Dracula is sleepy.     (Probable with d2)
```

Question: If Almost all > Most, should it be that d2>d1?

```
All vampires are sleepy.
Dracula is probably a vampire.
------------------------------
Dracula is sleepy.     (Probable with d3)
```

Question: d1 VS d3? d2 VS d3?

In general, if we define a notion of monotonicity in premises, would our notion of inference be compatible with them? That is, if premises of an argument A are stronger (by some measure of strength) than of an argument B, would the conclusion of B become more probable under the premises of A?

5.3 Probabilistic approach to pragmatics

5.3.1 The proviso problem

- If John goes to the sea he will take his cat with him. (Presupposition is that the cat and going to the sea are not related; so, John definitely has a cat.)
- If John goes to the sea then he will take his scuba. (Presupposition is that the scuba and the sea are related; so John doesn't need to have a scuba, but he might get one for this purpose.)

How to model this? Lassiter (2012) proposes to use probabilistic approach for that, which we adopt in our settings. Assume we have a candidate presupposition. If the candidate presupposition π is not probabilistically inferrable from the initial sentence, then we consider π to be an actual presupposition. Otherwise, π doesn't qualify for a presupposition.

5.3.2 The Rational Speech Act (RSA) model

Meaning is often interpreted **in contrast** to common knowledge. So how do we do this?

A popular approach to this kind of pragmatics is the RSA model. There is a whole ESSLLI course on this! We won't get into the details, but a quick outline is the following.

RSA assumes two agents, a listener $\mathscr{L}$ and a speaker $\mathscr{S}$. $\mathscr{S}$ utters a declarative sentence u heard by $\mathscr{L}$, without transmission error. The point of RSA is to model how, assuming Gricean cooperativeness between $\mathscr{S}$ and $\mathscr{L}$, $\mathscr{L}$ should disambiguate among possible interpretations of u.

1. Does u literally mean φ? For this we may use the model described above. (The uncertainty over the meaning of the utterance u is represented by a parameter θ.)

   $P_{L_0}(\varphi \mid u) = \mathbb{E}_{\theta \in Parameters}[[\![u]\!]^\theta \vdash \varphi]$

2. The (Gricean) speaker chooses u to (soft-)maximize the probability that $\mathscr{L}$ understands φ:

   $P_{S_1}(u \mid \varphi) \propto (P_{L_0}(\varphi \mid u)/C(u))^\alpha$

3. The (pragmatic) speaker considers the meaning of u in proportion to the estimated choices of the (Gricean) speaker.

   $P_{L_1}(\varphi \mid u) \propto P_{S_1}(u \mid \varphi) \times P(\varphi)$

Take away message: you can use an RSA layer **on top of** our probabilistic models.

1. Example

   S: "I ate 5 cookies".

   How many cookies did S eat?

   [[I ate n cookies]] = $\exists m \geq n\ \text{eat}_{\text{cookies}}(I, m)$

   If we consider only 3 possible utterances and meanings (say, there were 7 cookies in the box), we get the literal probability assignment $P_{L_0}(\varphi \mid u)$:

Utterance	ate(5)	ate(6)	ate(7)
I ate 5 cookies	1/3	1/3	1/3
I ate 6 cookies	0	1/2	1/2
I ate 7 cookies	0	0	1

softmax by column (α=4), assuming all utterances have the same cost, yields $P_{S_1}(u \mid \varphi)$:

Utterance	ate(5)	ate(6)	ate(7)
I ate 5 cookies	1	0.16	0.01
I ate 6 cookies	0	0.84	0.06
I ate 7 cookies	0	0	0.93

Normalizing by row yields $P_{L_1}(\varphi \mid u)$

Utterance	ate(5)	ate(6)	ate(7)
I ate 5 cookies	0.85	0.14	0.01
I ate 6 cookies	0	0.93	0.07
I ate 7 cookies	0	0	1

Exercises:

- iterate the process (to get a level-2 pragmatic listener)
- Repeat everything with α=∞
- Consider the utterance "I ate 2 cookies".

We won't further discuss the merits of this model.

In the above we have:

(a) used a "strict" probabilistic semantics

(b) and laid a pragmatic component on top of it.

BUT we can also **bake in** pragmatics into the *raw* probabilistic meaning.

For example, ⟦I ate n cookies⟧ = $\text{eat}_{\text{cookies}}$(I, m) where m is chosen in a Poisson distribution with mode n.

6 References

- Variational Inference: A Review for Statisticians (by David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe) `https://arxiv.org/pdf/1601.00670.pdf`
- Variational Inference (lecture notes by David M. Blei)

`https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf`

- A Tutorial on Variational Bayesian Inference `http://www.orchid.ac.uk/eprints/40/1/fox_vbtut.pdf`
- Bayesian inference `https://en.wikipedia.org/wiki/Bayesian_inference#Bayesian_inference`

- Dealing with observing zero-measure events Paradoxes of Probabilistic Programming And How to Condition on Events of Measure Zero with Infinitesimal Probabilities Jules Jacob, 2021
- Presuppositions, provisos, and probability Semantics & Pragmatics Volume 5, Article 2: 1–37, 2012 Lassiter, Daniel `https://semprag.org/index.php/sp/article/view/sp.5.2/pdf`
- PROBABILISTIC INFERENCE AND THE CONCEPT OF TOTAL EVIDENCE J. Hintikka & P. Suppes (Eds), Aspects of Inductive Logic, Amsterdam: North-Holland, 1966, pp. 49-65 PATRICK SUPPES `https://suppescorpus.sites.stanford.edu/sites/g/files/sbiybj7316/f/probabilistic_inference_and_the_concept_of_total_evidence_71.pdf`

6.1 Our references

- A Compositional {Bayesian} Semantics for Natural Language JP Bernardy, R Blanck, S Chatzikyriakidis, S Lappin Proceedings of the First International Workshop on Language Cognition and Computational Models
- Bayesian inference semantics: A modelling system and a test suite Jean-Philippe Bernardy, Rasmus Blanck, Stergios Chatzikyriakidis, Shalom Lappin, Aleksandre Maskharashvili Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM) 2019
- Predicates as Boxes in Bayesian Semantics for Natural Language JP Bernardy, R Blanck, S Chatzikyriakidis, S Lappin, A Maskharashvili Proceedings of the 22nd Nordic Conference on Computational Linguistics, 333-337
- Jean-Philippe Bernardy, Rasmus Blank, Aleksandre Maskharashvili, "A Logic with Measurable Spaces Fornatural Language Semantics", AMIM Vol.25 No.2, 2020, pp. 31-43
- Julian Grove, Jean-Philippe Bernardy, Stergios Chatzikyriakidis From compositional semantics to Bayesian pragmatics via logical inference NALOMA 2021