Logical Deduction

Preparation

It is recommended that you read the following chapters before doing this tutorial:

- Set
- Venn Diagram and Propositional Logic
- Logical Connectives and Propositional Equivalences

You should also be able to:

- Use list comprehension
- Use the Haskell functions below (as discussed in the logic lectures)

```
&& :: Bool -> Bool -> Bool
|| :: Bool -> Bool -> Bool
not :: Bool -> Bool
```

Google and Hoogle[1] are your friends. Post your questions on Piazza.

1 Sets and Predicates

To begin, we will continue to play with our universe of **Thing**s from the previous tutorial. The code from the previous tutorial is already pasted in the file Tutorial2.hs. We defined the `Thing` data type:

```
data Thing = R | S | T | U | V | W | X | Y | Z deriving (Eq, Show)
```

a list of all `Thing`s:

```
things = [R, S, T, U, V, W, X, Y, Z]
```

and 7 unary **predicates**:

```
isRed       :: Thing -> Bool
isBlue      :: Thing -> Bool
isGreen     :: Thing -> Bool
isDisc      :: Thing -> Bool
isTriangle :: Thing -> Bool
isSmall    :: Thing -> Bool
isBig      :: Thing -> Bool
```

A predicate is a function that takes some arguments and outputs a boolean. But instead of thinking about them as functions, we can also think about them as sets whose members we can list. Take our universe of `Thing`s for example, we can specify the set of red things by either,

- listing them

  ```
  [U, V]
  ```

- or selecting them using the predicate `isRed` in a list comprehension:

  ```
  GHCi> [x | x <- things, isRed x]
  [U,V]
  ```

  We test everything: if something is red, then it belongs to the set of red things.

We can establish the equality `isRed x = x `elem` [U, V]`. In this course, we define predicates as Boolean-valued functions, but we often picture the corresponding set, using the same name for both.

[1] https://hoogle.haskell.org/

2 Satisfaction

Once we have many (or even only a few) predicates, it is very natural for us to ask whether:

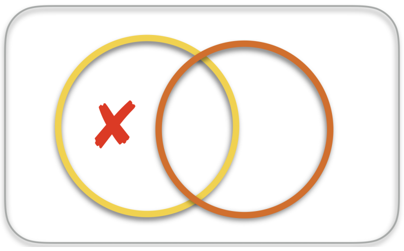
- some big disc is red.
- every big triangle is small.
- some big triangle is green.

You might have noticed the statements all share similar forms i.e. "**Every ... is ...**", "**Some ... is ...**" etc. We will see that four forms arise naturally:

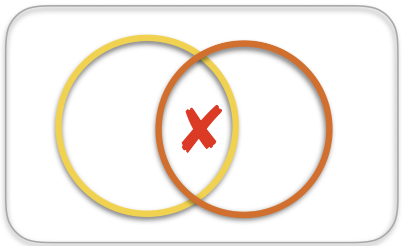
- Every a is b.
- No a is b.
- Some a is b.
- Some a is not b.

We can use Venn diagrams to visualize these 4 statements and your task is to do exactly that. The Venn diagram for two predicates divides the universe into four regions. Fill in a Venn diagram for each of the statements (the first one is filled for you), put a tick in every region that is certainly not empty and a cross in every region that is certainly empty.

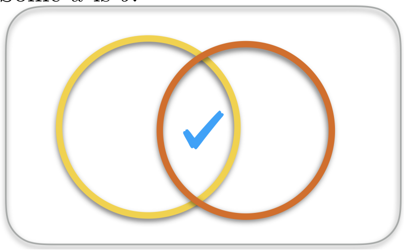
1. Every a is b.



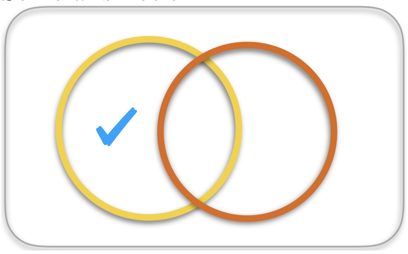
2. No a is b.



3. Some a is b.



4. Some a is not b.

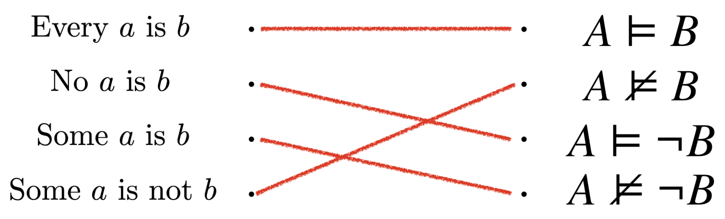


To make the logic of the situation clear it will help to avoid natural language, which is often ambiguous.

We start from the inclusion relationship, **every** a **is** b, which says that the set of individuals for which a is true is included in the set of individuals for which b is true. We write this as $a \vDash b$ (reads **every** a **satisfies** b, or just, a **satisfies** b).

Clearly in any universe, given two predicates a and b, either, $a \vDash b$ will be true, or it will be false, which means that **some** a **does not satisfy** b, which we write as $a \nvDash b$. But for different predicates or in a different universe the answer may be different.

5. Connect each of the following natural language statements with the corresponding symbolic form.



Aside: What does it mean to negate a predicate?

Negation of Boolean values is easy, 'not True is False' and 'not False is True'. In Haskell, the function `not :: Bool -> Bool` implements this operation. But we know that predicates are functions of type `U -> Bool` (`U` stands for Universe). Negations of predicates should still be predicates, so the type of a function that negates predicates should be:

```
neg :: (Thing -> Bool) -> (Thing -> Bool)
```

The things satisfying `a` should be the things not satisfying `neg a`. We implement the function `neg` to do exactly that:

```
(neg a) x = not (a x)
```

At first glance, it might feel like this function is rather pointless, but it is a first example of a very important general construction.

6. Implement the infix function `(|=) :: (Thing -> Bool) -> (Thing -> Bool) -> Bool` in Haskell so that `a |= b` returns `True` if and only if, in our toy universe, every *a* is *b*.

Solution:

```
(|=) :: (Thing -> Bool) -> (Thing -> Bool) -> Bool
(|=) a b = and [b x | x <- things, a x]
```

7. Implement the infix function `(|/=) :: (Thing -> Bool) -> (Thing -> Bool) -> Bool` in Haskell so that `a |/= b` returns `True` if and only if, in our toy universe, some *a* is not *b*.

Solution:

```
(|/=) :: (Thing -> Bool) -> (Thing -> Bool) -> Bool
(|/=) a b = or [neg b x | x <- things, a x]
```

To verify the correctness of your implementation[2], run in GHCi:

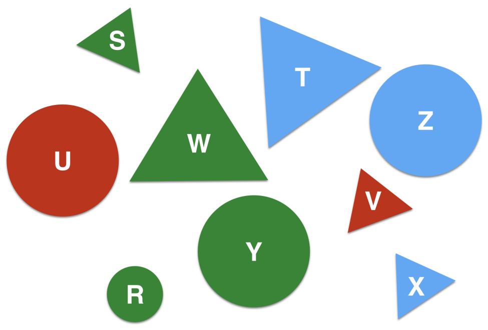
```
GHCi> quickCheck propEntail
+++ OK, passed 100 tests.
GHCi> quickCheck propNotEntail
+++ OK, passed 100 tests.
```

If you get messages like the ones below, then your implementation is not quite correct.

```
GHCi> quickCheck propEntail
*** Failed! Falsifiable (after 1 test):
Sorry, your implementation is not quite correct:(
Sorry, your implementation is not quite correct:(
```

[2]It is very important for you to get question 6 and 7 correct, otherwise follow-up exercises won't work.

For each of the following satisfaction relations, determine whether it is valid. First figure it out yourself, then use Haskell to verify.



8. `isRed |= neg isGreen`
9. `neg isGreen |= isRed`
10. `neg isDisc |= isTriangle`
11. `isTriangle |= neg isDisc`

We can make things more interesting by adding "or" and "and" operators. We again need to think about what it means to take the disjunction or conjunction of two predicates.

12. Implement a version of "or" `(|:|)` and "and" `(&:&)` for predicates.

Solution:

```
(|:|) :: (Thing -> Bool) -> (Thing -> Bool) -> (Thing -> Bool)
(|:|) a b x = a x || b x

(&:&) :: (Thing -> Bool) -> (Thing -> Bool) -> (Thing -> Bool)
(&:&) a b x = a x && b x
```

Continue and determine whether the following satisfaction relations are valid, use Haskell to verify.

13. `isBig &:& isRed |= isDisc`
14. `isBig &:& isDisc |= isRed`
15. `isSmall &:& neg isGreen |= neg isDisc`
16. `isBig |:| isGreen |= neg isTriangle`
17. `neg (isTriangle |:| isGreen) |= isDisc`
18. `neg isTriangle &:& neg isGreen |= isDisc`

Are there any connections between 17 and 18?

3 Syllogisms

A syllogism is a kind of logical argument that applies deductive reasoning to arrive at a conclusion based on two or more propositions that are asserted or assumed to be true.[3]
Here is an example:

$$\begin{array}{rl} & \text{Every human is mortal.} \\ & \text{Every Greek is humans.} \\ \therefore & \text{Every Greek is mortal.} \end{array}$$

The $\therefore$ sign reads **therefore**. The first two statements are called **premises** and the last is called **conclusion**. It should now be clear that we can translate each line into a satisfaction relation:

$$\begin{array}{rl} & \text{Human} \vDash \text{Mortal} \\ & \text{Greek} \vDash \text{Human} \\ \therefore & \text{Greek} \vDash \text{Mortal} \end{array}$$

We will separate the **premises** from the **conclusion** by putting the premises above and the conclusion beneath a line. The line is called an **inference** line and we say that the premises **infer** the conclusion:

$$\frac{\text{Human} \vDash \text{Mortal} \qquad \text{Greek} \vDash \text{Human}}{\text{Greek} \vDash \text{Mortal}}$$

This syllogism is **sound**, meaning that the conclusion does indeed follow from the premises. Below is an **unsound** syllogism:

$$\frac{\text{Some food is red.} \qquad \text{Some plant is food.}}{\text{Some plant is red.}}\ \text{unsound}$$

A **counterexample** would be a universe in which some food is red, some plant is food, and no plant is red. The premises hold in this universe but not the conclusion.

Lewis Carroll, the author of *Alice's Adventures in Wonderland* and *Through the Looking Glass*, taught logic at Oxford in the 1890s. In his words,

> When the terms of a proposition are represented by *words*, it is said to be in **concrete** form; when by *letters*, **abstract**.

To translate a proposition from concrete to abstract form, we must choose a universe, and regard each term as a predicate, to which we assign a letter. He gives this example:

$$\begin{array}{rll} & \text{All cats understand French} & m \vDash x \\ & \text{Some chickens are cats.} & y \nvDash \neg m \\ \therefore & \text{Some chickens understand French.} & y \nvDash \neg x \end{array}$$

where universe = animals, m = cats, y = chickens, x = understanding French. We can put this (sound) argument into inference form:

$$\frac{m \vDash x \qquad y \nvDash \neg m}{y \nvDash \neg x}$$

[3] https://en.wikipedia.org/wiki/Syllogism

Translate the following arguments, examples from Lewis Carroll's book[4] and Michael Curthbert's blog[5], into syllogistic form. Try to determine, without using venn diagrams or formal logic, which of them are instances of logically sound rules.

1. All diligent students are successful.
   All ignorant students are unsuccessful.
   ∴ Some diligent students are ignorant.

Solution: $\dfrac{d \vDash s \quad i \vDash \neg s}{d \nvDash \neg i}$ where
$d =$ diligent students
$i =$ ignorant students
$s =$ successful students

2. All soldiers can march.
   Some babies are not soldiers.
   ∴ Some babies cannot march.

Solution: $\dfrac{s \vDash m \quad b \nvDash s}{b \nvDash m}$ where
$s =$ soldiers
$m =$ people who can march
$b =$ babies

3. No Professors are ignorant.
   All ignorant people are vain.
   ∴ No professors are vain.

Solution: $\dfrac{p \vDash \neg i \quad i \vDash v}{p \vDash \neg v}$ where
$p =$ professors
$i =$ ignorant people
$v =$ vain people

4. Every eagle can fly.
   Some pigs cannot fly.
   ∴ Some pigs are not eagles

Solution: $\dfrac{e \vDash f \quad p \nvDash f}{p \nvDash e}$ where
$e =$ eagles
$f =$ things that can fly
$p =$ pigs

5. No pets are furry beasts.
   Some rabbits are pets.
   ∴ Some rabbits are not furry beasts.

Solution: Question 9 is the solution where
$m =$ pets
$p =$ furry beasts
$s =$ rabbits

6. No insects are delicious foods.
   All tacos are delicious foods.
   ∴ No tacos are insects.

Solution: Question 10 is the solution where
$m =$ insects
$p =$ delicious food
$s =$ tacos

7. Some women are humans.
   Some men are women.
   ∴ Some men are humans.

Solution: Question 11 is the solution where
$m =$ women
$p =$ humans
$s =$ men

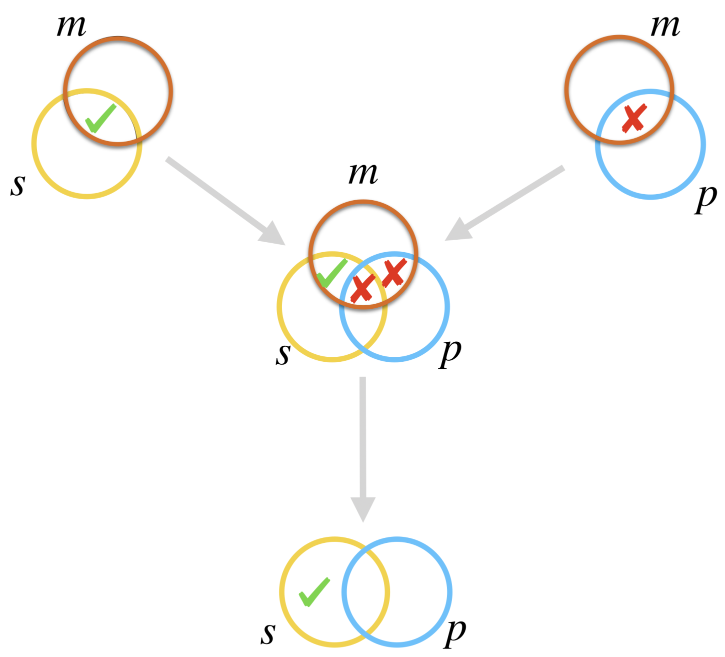
8. All furry beasts are pets.
   No rabbits are pets.
   ∴ Some rabbits are not furry beasts.

Solution: Question 12 is the solution where
$m =$ pets
$p =$ furry beasts
$s =$ rabbits

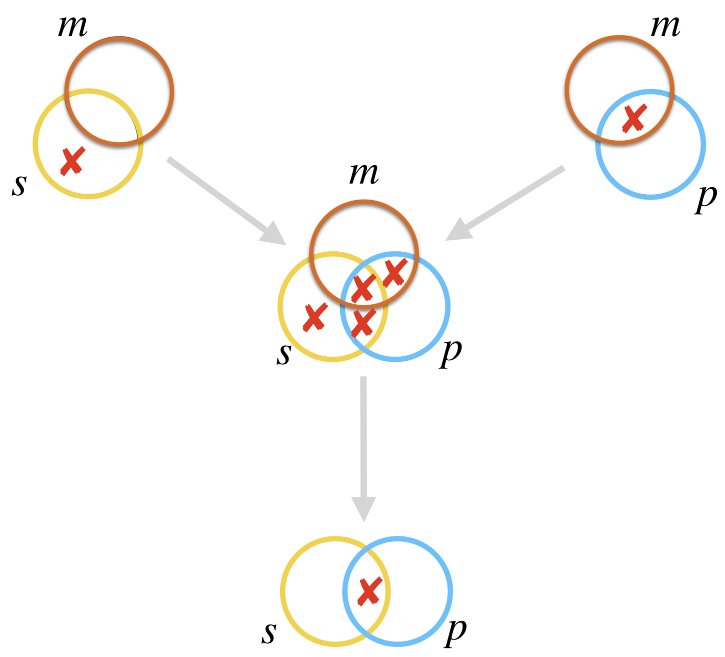
[4] Symbolic Logic
[5] http://prolatio.blogspot.com/2016/08/all-256-syllogism-forms-with-examples.html

For each of the syllogisms below, determine whether it is sound using a Venn diagram.[6] If it is not, give a counterexample.

9. $\dfrac{m \vDash \neg p \qquad s \nvDash \neg m}{s \nvDash p}$

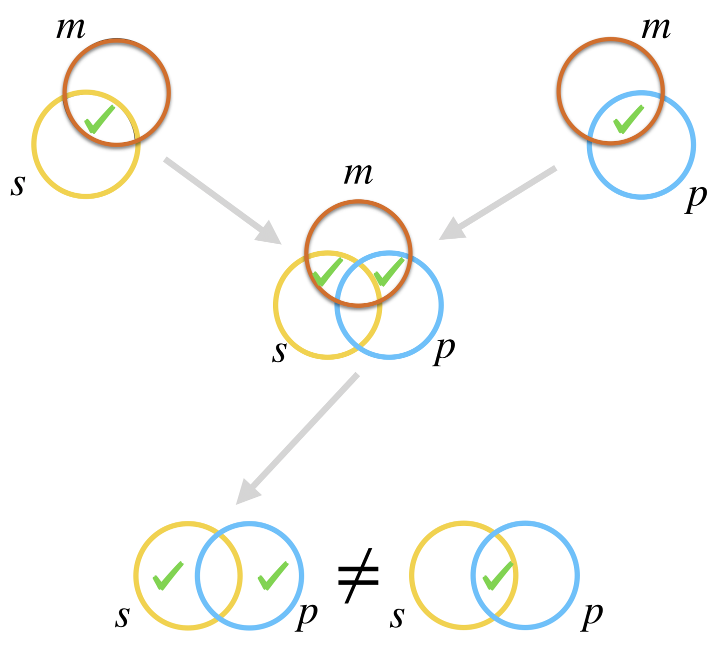


10. $\dfrac{m \vDash \neg p \qquad s \vDash m}{s \vDash \neg p}$



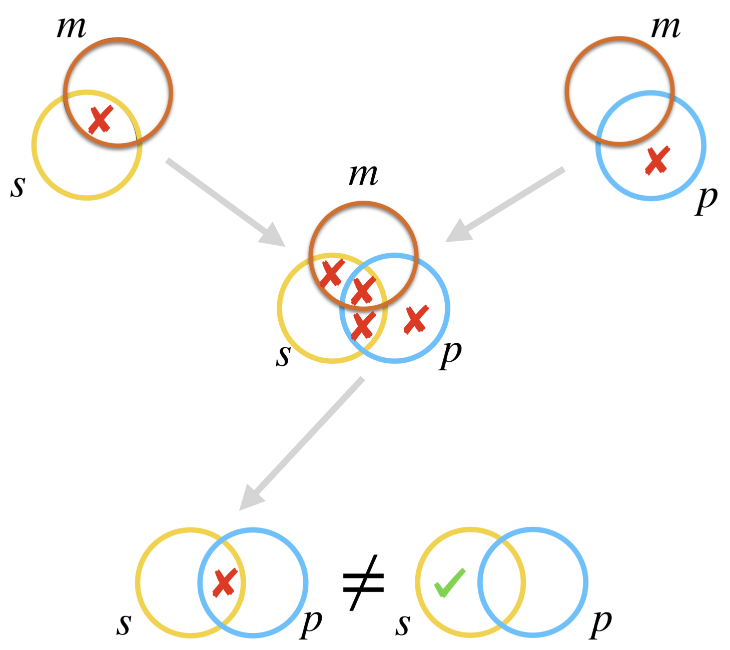
[6]Tick the non-empty region(s) and cross the empty region(s).

11. $\dfrac{m \nvDash \neg p \qquad s \nvDash \neg m}{s \nvDash \neg p}$



Counterexample: Something is $s \wedge m$, something is $m \wedge p$, nothing is $s \wedge p$.
The premises still hold, but not the conclusion.

12. $\dfrac{p \vDash m \qquad s \vDash \neg m}{s \nvDash p}$



Counterexample: Nothing is s, nothing is $p \wedge \neg m$.
The premises still hold, but not the conclusion.

13. We have shown in class that valid syllogisms may be derived from other valid syllogisms using substitution, double negation elimination, and two kinds of contraposition. The two forms of contraposition are:

    - Contraposition of the antecedent and succedent in a sequent of the form $a \vDash \neg b$, or the same form negated $a \nvDash \neg b$.
    - Contraposition of the conclusion of a rule with either one of its premises.

    Consider the syllogism

    Some holidays are rainy.
    Rainy days are tiresome.
    ∴ Some holidays are tiresome.

    Put this in symbolic form where universe = days, h = holidays, r = rainy days, t = tiresome days. Derive as many valid syllogisms (with double negations eliminated) as you can from this example using a single step of substitution or contraposition.

    Solution: 5. 1 with substitution, 2 with sequent contraposition, 2 with rule contraposition

14. What can you say about the number of occurrences of $\nvDash$ in any sound syllogism? What can you say about the number of occurrences of $\neg$ in any sound syllogism?

    Consider the following syllogism,

    No animals are unicorns.
    All unicorns are horses.
    ∴ Some horses are not animals.

    Translate it to symbolic form, and explain, without further calculation, and without giving a counterexample, why it is not sound.

    Solution: An even number (0 or 2) of occurrences for each of $\nvDash$ and $\neg$.

A challenge

Aristotle considered some syllogisms some syllogisms to be sound that we now consider unsound! This is not because he made a mistake, but because he made an assumption that we do not make. For example, Aristotle thought that, when we say that, *every a is b*, we implicitly make the so-called **existential assumption**, that, *there is some a* , so that it follows that, *some a is b*.

15. To say that there is some a, we can say, some a is a. Your first task is to write this symbolically in terms of $\vDash$.

    Solution: $a \nvDash \neg a$

16. Your second task is to find an example among the unsound examples in the preceding questions that is sound if we add the existential assumption. You can show this either by an argument using Venn diagrams or by linking rules together to make a formal deduction.

    Solution: question 8 is sound if we add existential assumption.

17. Final task: We have identified 15 sound syllogisms without the existential assumption. Using the existential assumption, Aristotle found nine more. Can you find them all?

    Solution: See Wikipedia "Syllogisms" for an exhaustive (classical) treatment.