II: Propositional logic
=======================

Examples:

Socrates is a man or Socrates is a woman.
Socrates is not a woman.
Therefore: Socrates is a man.

If Socrates is a vampire and vampires are immortal, then Socrates is alive.
Socrates is not alive.
Therefore: Either Socrates is not a vampire, or vampires are not immortal.

We will develop a formal system, the _propositional **calculus_**, implementing this kind of logic.

Our system ***won't*** have strings interpreted as "Socrates" or "is a vampire" (we'll have to wait for the predicate calculus for that!). Rather, we use _propositional variables_ to stand in for whole propositions - e.g. **P** could stand for "Socrates is a vampire". For our purposes, a _proposition_ is just something which is true or false.

The language of propositional logic

Alphabet: **<**, **>**, **P**, **Q**, **R**, ', /\, \/, **=)**, ~
(Note: I'm using "**=)**" as an ascii representation of the horseshoe character)

Well-formedness:

Well-formed strings in propositional logic are called _well-formed formulas_ (wffs).

Rules to determine well-formedness:

* "**P**", "**Q**" and "**R**" are well-formed, as are "**P'**", "**R''**" and so on. These are the _propositional **variables_**. We also refer to them as **_atoms_**.
* If **x** is a wff, then **~x** is a wff
* If **x** and **y** are wffs, then **<x /\ y>**, **<x \/ y>** and **<x =) y>** are wffs.
* Nothing else is a wff!

Unique readability:

A wff is of _precisely one_ of the forms given above, so we can tell exactly how it was built up from variables. This is called _parsing_ the wff, and we can draw the result as a _parse **tree_**.

For example, the wff **\sigma = "<<P /\ ~Q> =) ~<<P \/ R> /\ P'>>**" has the following parse tree:

```
                  \sigma
                   / \
                  /   \
                 /  =)  \
           <P /\ ~Q>      ~<<P \/ R> /\ P'>>
              / \                  |
             /   \                 | ~
            /  /\  \               |
           P        ~Q      <<P \/ R> /\ P'>>
                     |              / \
                     | ~           /   \
                     |            /  /\  \
                     Q      <P \/ R>      P'
                               / \
                              /   \
                             /  \/  \
                            P        R
```

Digression:

Contrast with natural languages, where parses are often not unique - sentences are often syntactically **_ambiguous_**.
e.g. "pretty little girls' school" has many parses (a school for girls which is quite little? **A** school owned by girls who are small and pretty?

etc)

Interpretations

Suppose we have an interpretation of the propositional variables
(e.g. **P -->** "Socrates is a vampire" etc). We extend the interpretation to
determine truth of arbitrary wffs by requiring, for **x** and **y** wffs:

- **~x** is true iff **x** is false;
- **<x /\ y>** is true iff **x** and **y** are both true;
- **<x \/ y>** is true iff at least one of **x** and **y** are true;
- **<x =) y>** is ***false*** iff **x** is true and **y** is false.

Since every wff has a unique parse, these rules decide the truth of every
wff.

Example:
According to an interpretation in which **P** and **Q** are true but **Q** and **P'** is
false, determine from the parse tree whether **\sigma** is true.

So
~ **-->** "not"
/ **-->** "and"
\/ **-->** "or"
=) **-->** "implies", "if [...] then [...]"

Regarding "or":
In English, "or" is sometimes _inclusive_
("Don't touch anything which is hot or which has sharp points!"
applies to things which are hot _and_ have sharp points)
and sometimes _exclusive_
(e.g. "a person is either male or female"
makes the (contentious!) claim that no-one can be both or neither)
("either" is mostly needed to clearly signal an exclusive or in
english);
<x \/ y> --> "**x** or **y**" in the ***inclusive*** sense.

Regarding "if":
It seems we are declaring that "if **P** then **Q**" is false iff **P** is true and **Q**
is false.

e.g. "If 4 is prime then there is a god" is true!

Consider:
"For every natural number **n**, if **n** is prime then **n=2** or **n** is odd." (*****)

This is true, precisely because:
for those **n** for which "**n** is prime" is true, "**n=2** or **n** is odd" is true.

For **n** for which **n** is ***not*** prime, "**n=2** or **n** is odd" is sometimes true and
sometimes false.

So in other words, (*****) is true precisely because
for all **n**, **<"n** is prime" **=) <"n=2 \/ "n** is odd"**>>** is true.

Digression:
What about natural language conditionals?
"If I had a million dollars, then I would be guilty of theft."

We can analyse this as
"For all imaginable situations **s**: if I have a million dollars in
s, then I am guilty of theft in **s**"

So is "if 4 were prime, then there would be a god" true? Not if it's
imaginable that 4 is prime and there is no god!

Tautologies, contradictions and satisfiability

Definition:
A _truth assignment_ is an assignment of a _truth **value_**, True or False,
to each propositional variable.

As above, a truth assignment determines truth values for all wffs.

// Truth assignments are the austere cousins of interpretations - we
// explicitly don't care about giving any "meaning" to the variables, we just
// give them truth values.

Definition:
A wff is a _tautology_ if it is True for every truth assignment.

A wff is a _contradiction_ if it is False for every truth assignment.

A wff is _satisfiable_ if it is not a contradiction, i.e. if it is True for some truth assignment.

Examples:
<P \/ ~P> is a tautology
<P /\ ~P> is a contradiction
<P =) ~P> is satisfiable, but not a tautology

Remark:
x is a contradiction iff **~x** is a tautology.
x is satisfiable iff **~x** is not a tautology.

Remark:
There is a decision procedure for being a tautology:
Given a wff **\sigma**, only finitely many propositional variables occur in **\sigma.**
For each possible assignment of True and False to those propositional variables, follow the parse tree of **\sigma** to determine whether **\sigma** is assigned True or False.
\sigma is a tautology iff it is True for all such truth assignments.

Similarly, we can decide being a contradiction and being satisfiable.

Note that if **n** different propositional variables occur in **\sigma**, we must check **2^n** assignments.

Truth tables

// Truth tables give a neat way to write down the above algorithm.

Truth table for the basic logical operators:

P	**Q**	**<P /\ Q>**	**<P \/ Q>**	**<P =) Q>**	**~P**
T	**T**	**T**	**T**	**T**	**F**
T	**F**	**F**	**T**	**F**	**F**
F	**T**	**F**	**T**	**T**	**T**
F	**F**	**F**	**F**	**T**	**T**

Truth table for **\sigma := <<~P =) <Q /\ R>> =) <<~R \/ ~Q> =) P>>**

P	**Q**	**R**	**~P**	**~Q**	**~R**	**<Q/\R>**	**<~P=)<Q/\R>>**	**<~R\/~Q>**	**<<~R\/~Q>=)P>**	**\sigma**
T	**T**	**T**	**F**	**F**	**F**	**T**	**T**	**F**	**T**	**T**
T	**T**	**F**	**F**	**F**	**T**	**F**	**T**	**T**	**T**	**T**
T	**F**	**T**	**F**	**T**	**F**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**T**	**T**	**T**	**T**
F	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**T**	**F**	**T**
F	**T**	**F**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**T**	**F**	**T**
F	**F**	**F**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**T**

So **\sigma** is a tautology.

Example Zen interpretation (after Hofstadter):
P --> "You are close to the way"
Q --> "This mind is Buddha"
R --> "The flax weighs three pounds"
\sigma --> "If your not being close to the way implies that this mind is Buddha and this flax weighs three pounds, then you are close to the

way if this mind is not Buddha or this flax does not weigh three pounds".
\sigma has truth-nature.

Notation:
We write **|=\sigma** to mean that **\sigma** is a tautology.

Remark:
Tautologies of the form **<\tau =) \theta>** express _valid **reasoning_**: whatever propositions the variables stand for, if **\tau** is true then **\theta** is true.

Exercise:
The decision procedure for tautologicalness of a wff **\sigma** described above requires us to check each of **2^n** truth assignments, where **n** is the number of variables appearing in **\sigma**.

Find a more efficient algorithm - one which, for some **c** and **k**, takes at most **cn^k** cpu cycles to run. Alternatively, prove that no such algorithm exists.

Note that you've determined whether **P=NP**, solving the most important problem in computer science. Claim plaudits, prizes, fame, and 7 RPs.

Example:
Using truth tables to solve a Smullyan-style knight-knave puzzle.

You are lost in a maze on Smullyan Island. Each inhabitant of this strange island is either a _knight_ or a **_knave_**. Everything a knight says is true, while everything a knave says is false.

Walking along a corridor while trying to find the way out, you come across an inhabitant of the island. You ask him for directions, and he says "If I am a knight, then the exit lies behind me".

Should you continue past him?

Solution:
Write **P** for the proposition "The inhabitant is a knight".
Write **Q** for the proposition "The exit is past the knight".
So the inhabitant is claiming
\sigma := <P =) Q> .
So **\sigma** is true iff the inhabitant is a knight; i.e. we know that
<<P =) \sigma> /\ <\sigma =) P>> is true.
Now write a truth table, and see what this being true tells us about Q's truth value.

A formal system for propositional logic
=======================================

We develop a formal system, PROP, to capture tautologies:
\sigma will be a theorem of PROP iff **|=\sigma**.

[We follow Hofstatder, Ch. VII. It's a Fitchish natural deduction system]

Alphabet:
The alphabet of propositional logic, with two new symbols '[' and ']'.
Axioms:
None!
Production Rules:
Joining:
(**x**, **y**) **|-> <x /\ y>**
Separation:
<x /\ y> |-> x
<x /\ y> |-> y
Double-Tilde:
~~x |-> x
x |-> ~~x
Detachment:
(**x**, **<x =) y>**) **|-> y**
Contrapositive:
<x =) y> |-> <~y =) ~x>
<~x =) ~y> |-> <y =) x>

```
    De Morgan:
        <~x /\ ~y> |-> ~<x \/ y>
        ~<x \/ y> |-> <~x /\ ~y>
    Switcheroo:
        <x \/ y> |-> <~x =) y>
        <~x =) y> |-> <x \/ y>
```

// No axioms so no theorems!
// That's because we're missing the informal rule!

Fantasy rule

At any point during a derivation, we may "push into a fantasy":
we write "[" on a line, and then ***any*** wff **x** on the next line.
We then proceed as if this is an entirely new derivation. Say we derive **y**.
We may then "pop out of the fantasy":
we write "]" on the line after **y**, and then "**<x =) y>**" on the line after that, and proceed as if the fantasy never happened (no lines from a popped fantasy may be used in production rules).

Example:

```
    [
        P       (pushing in to a fantasy)
        ~~P     (double-tilde)
    ]
    <P =) ~~P>  (fantasy rule)
```

So the fantasy rule implements the reasoning
"if from **x** we can prove **y**, then **<x =) y>** must be true".

Note we may push into a new fantasy within a fantasy, and we must pop out of the inner fantasy before popping out of the outer fantasy (indentation helps to keep track!).

Example:

```
    [
        <<P =) P> =) Q>
        [
            P
        ]
        <P =) P>            (fantasy)
        Q                   (detachment)
    ]
    <<<P =) P> =) Q> =) Q>  (fantasy)
```

Carry-over rule: inside a fantasy, we may write any line which appeared in the "reality one level up".

Example:

```
    [
        P
        [
            Q
            P               (carry-over)
            <Q /\ P>        (joining)
        ]
        <Q =) <Q /\ P>>     (fantasy)
    ]
    <P =) <Q =) <Q /\ P>>>  (fantasy)
```

Whee!

Remark:
Please note that by introducing this rule, we've broken the feature of our previous systems that every line of a derivation is a theorem. With the fantasy rule, ***any*** wff can appear as a line! The theorems are the wffs on lines which aren't part of any fantasy (i.e. the unindented lines, if we indent as above).

Notation:
We write "**|- \sigma**" to mean that **\sigma** is a PROP-theorem.

Waiter, waiter, there's an informal rule in my formal system!

Don't worry!

Fact: We can find a Post formal system, in the strict sense we've been using, which has the same theorems as the system described above.

How to do that (omitted in class):

Actually, there are two sensible ways to do this. The traditional approach would be to scrap the natural deduction scheme described above, and instead use a Hilbert-style deduction system. In these, the only rule of inference is detachment ("modus ponens"), and we have some well-chosen axiom schemes. You can look this up if you're interested.

But we don't need to do that. We can implement the fantasy rule directly in syntax. Here's a way to do that; the basic idea is just to keep track of the premises of the fantasies we're inside:

```
Alphabet: as above, but add new symbols |- ? W F :
Axioms: |-, WFF:P, WFF:Q, WFF:R
Production rules:
    (x|-y, WFF:z)  |->  x?z|-z      (pushing into a fantasy)
    (x|-y, WFF:z)  |->  x?z|-y      (carry-over)
    (x?y|-z, WFF:y) |->  x|-<y=)z>  (popping out of a fantasy)

    x|-<y/\z> |-> x|-y
    x|-<y/\z> |-> x|-z
    (x|-y, x|-z) |-> x|-<y/\z>
        and so on for the other rules in the original system

    WFF:Px  |->  WFF:P'x
    WFF:Qx  |->  WFF:Q'x
    WFF:Rx  |->  WFF:R'x            (variables are well-formed)

    WFF:x   |->  WFF:~x
    (WFF:x, WFF:y) |-> WFF:<x/\y>
    (WFF:x, WFF:y) |-> WFF:<x\/y>
    (WFF:x, WFF:y) |-> WFF:<x=)y>   (formation rules for wffs)

    |-x  |->  x                     (deriving wffs)
```

The last example of the previous section, derived in this system:

```
|-
WFF:P
?P|-P
WFF:Q
?P?Q|-Q
?P?Q|-P
?P?Q|-<Q/\P>
?P|-<Q=)<Q/\P>>
|-<P=)<Q=)<Q/\P>>>
<P=)<Q=)<Q/\P>>>
```

Examples

Give derivations of the following tautologies.

```
<<P =) Q> =) <~Q =) ~P>>
    (contraposition)
<P \/ ~P>
    ("excluded middle")
<<P /\ ~P> =) Q>
    (you can prove anything from a contradiction!)
~<P /\ ~P>
    Hint: first prove <<P /\ ~P> =) ~<P \/ ~P>>
<<P =) <Q /\ ~Q>> =) ~P>
    (proof by contradiction)
<<<P \/ Q> /\ <<P =) R> /\ <Q =) R>>> =) R>
    (cases)
<~<P /\ Q> =) <~P \/ ~Q>>
    (more De Morgan)
<<<P =) <P =) Q>> /\ <<P =) Q> =) P>> =) <P /\ Q>>
```

(cf knight-knave example above)

Substitution

Definition:
Let **\sigma** be a wff, and let **p_1**, ..., **p_n** be propositional variables
appearing in **\sigma**. Let **\phi_1**, ..., **\phi_n** be wffs. Then if we replace
each occurence of **p_i** in **\sigma** with **\phi_i**, we get a new wff. Such a wff
is called a _substitution instance_ of **\sigma**.
Lemma: Suppose **\tau** is a substitution instance of **\sigma**. Then
(a) if **|= \sigma** then **|= \tau**
(**b**) if **|- \sigma** then **|- \tau**
Proof:
(a) Exercise
(**b**) Make the substitution throughout a derivation of **\sigma**; the result is
also a derivation.

Example:
We saw that
|- <P \/ ~P>.
So by substituting **<P =) Q>** for **P**, it follows that
|- <<P =) Q> \/ ~<P =) Q>>.
Similarly for **|=**.

Soundness

Theorem [Soundness]:
For any wff **\tau**, if **|- \tau** then **|= \tau**.

Lemma:
The production rules correspond to tautologies:
|= <<P/\Q> =) P> **(separation)**
|= <<P /\ <P =) Q>> =) Q> **(detachment)**
|= <<P /\ Q> =) <P /\ Q>> **(joining)**
|= <~~P =) P> **(double-tilde)**
etc
Proof:
Check truth tables. Exercise.

We would like now to prove the theorem by induction on the length of a
derivation - but the induction hypothesis tells us nothing about lines which
occur within fantasies...

Definition:
A set of wffs **\Sigma _necessitates_** a wff **\tau**, written
\Sigma |= \tau,
if **\tau** is true for all truth assignments for which every **\sigma** in **\Sigma**
is true.

|= \tau abbreviates **\emptyset |= \tau**.

[Hoping to avoid giving this, as it just seems obfuscatory
Notation:
(just to clarify)
Recall that a truth assignment is a map
f : {propositional variables} -> {T,F} .
Write **f*** for the unique extension
f* : {wffs} -> {T,F}
such that for all wffs **\sigma,\tau**:
f*(~\sigma)=T iff **f*(\sigma)=F**,
f*(<\sigma /\ \tau>)=T iff **f*(\sigma)=T=f*(\tau)**,
f*(<\sigma \/ \tau>)=F iff **f*(\sigma)=F=f*(\tau)**,
and **f*(<\sigma =) \tau>)=F** iff **f*(\sigma)=T** and **f*(\tau)=F**.

(so "**\sigma** is true for **f**" means **f*(\sigma)=T**).

Then we can write the definition of **\Sigma |= \tau** more formally as:
for all **f**, if **f*(\sigma)=T** for all **\sigma \in \Sigma** then **f*(\tau)=T**.
]

Definition:
The _premise_ of a fantasy is its first line.

The _premises of a line_ of a PROP-derivation are the premises of the fantasies the line appears within.

Claim:
Let **\tau** be a wff occuring as a line of a PROP-derivation.
Let **\Sigma** be the set of premises of the line.
Then **\Sigma |= \tau**.

Proof:
Assume the claim holds for the first **k** lines of any derivation, we show it holds for the first **k+1**. So suppose the (**k+1**)th line of a derivation is a wff **\tau** with premises **\Sigma**.

If **\tau** is the premise of a fantasy, then **\tau \in \Sigma**, so clearly **\Sigma |= \tau**.

If **\tau** is a carry-over, then **\tau** appears as a previous line with premises **\Sigma'** a subset of **\Sigma**; by the inductive hypothesis, **\Sigma' |= \tau**, so also **\Sigma |= \tau**.

If **\tau** is the result of the fantasy rule, then **\tau = <\phi =) \psi>**, and **\psi** appears on a previous line with premises **\Sigma \union {\phi}**, so by the inductive hypothesis
\Sigma \union {\phi} |= \psi.
Now for any truth assignment for which all **\sigma\in\Sigma** are true:
if **\phi** is true then **\psi** is true since **\Sigma \union {\phi} |= \psi**;
hence **\tau = <\phi =) \psi>** is true.
So **\Sigma |= \tau**.

[Phrasing that argument with the fs:
Now let **f** be a truth assignment, and suppose **f*(\sigma)=T** for all **\sigma \in \Sigma**. Suppose **f*(<\phi =) \psi>)=F**. Then **f*(\phi)=T** and **f*(\psi)=F**, contradicting **\Sigma \union {\phi} |= \psi**. So **f*(<\phi =) \psi>)=T**. So **\Sigma |= \tau**.
]

Else, **\tau** is the result of a production rule. Say it has two inputs, **\phi** and **\psi**. Each appears as a previous line in the derivation with the same premises **\Sigma**, so by the inductive hypothesis,
\Sigma |= \phi and **\Sigma |= \psi**.
By the Lemma,
|= <<\phi /\ \psi> =) \tau>.
It follows easily that **\Sigma |= \tau**.

(if the production rule has only one input, the argument is similar)

Completeness

Definition:
For a set **\Sigma**, write
\Sigma |- \tau **("\Sigma proves \tau")**
to mean **\tau** is a theorem of the system **PROP+\Sigma** we get by adding **\Sigma** as axioms to PROP.

Lemma ["strong soundness"]:
If **\Sigma |- \tau** then **\Sigma |= \tau**
Proof:
Suppose **\Sigma |- \tau**. So there is a derivation of **\tau** using **\Sigma** as axioms. The derivation can use only finitely many of the axioms, say **\sigma_1**, ..., **\sigma_n**. Let **\phi** be the conjunction
\phi := <\sigma_1 /\ <\sigma_2 /\ ... /\ \sigma_n>...>>

Then by separation and the fantasy rule,
|- <\phi =) \tau>.

By soundness,
|= <\phi =) \tau>.

It follows easily that
\Sigma |= \tau.

```
Lemma 1:
    For each of ~, /\, \/, =), the tautologies corresponding to its truth
    table are theorems; i.e.
        <P/\Q>:
            |- <<P/\Q> =) <P/\Q>>
            |- <<~P/\Q> =) ~<P/\Q>>
            |- <<P/\~Q> =) ~<P/\Q>>
            |- <<~P/\~Q> =) ~<P/\Q>>
        ~P:
            |- <P =) ~~P>
            |- <~P =) ~P>
    and similarly for \/ and =).
Proof:
    All fairly straightforward. See exercises.

Lemma 2:
    |- <<<P =) Q> /\ <~P =) Q>> =) Q>
Proof:
    Here's a PROP-derivation:
    [
        <<P =) Q> /\ <~P =) Q>>
        <P =) Q>
        <~P =) Q>
        [
            ~Q
            <P =) Q>
            <~Q =) ~P>
            ~P
            <~P =) Q>
            Q
        ]
        <~Q =) Q>
        <Q \/ Q>
        [
            ~Q
            <~Q /\ ~Q>
            ~<Q \/ Q>
        ]
        <~Q =) ~<Q \/ Q>>
        <<Q \/ Q> =) Q>
        Q
    ]
    <<<P =) Q> /\ <~P =) Q>> =) Q>


Theorem [completeness of PROP]:
    For any wff \tau, if |= \tau then |- \tau.

Proof:
    Notation:
        If PV = {p_1, ..., p_n} is a set of propositional variables and
        f : PV -> {T,F}, write
            \Sigma^f := { +/- p_i | 1 <= i <= n }
        where +/- p_i = p_i if f(p_i)=T, and +/- p_i = ~p_i if f(p_i)=F.
    Claim:
        If \sigma is a wff and all propositional variables occuring in \sigma
        are in PV, then for any f,
            \Sigma^f |- \sigma  or  \Sigma^f |- ~\sigma  (*)
    Proof:
        By induction on depth of \sigma's parse tree.

        If \sigma is a propositional variable, (*) is clear.

        Else, clear by Lemma 1 and the inductive hypothesis.

    Now let PV = {p_1, ..., p_n} be the set of propositional variables
    occuring in \tau.

    So by the claim, "strong soundness" and the fact that \tau is a tautology,
    for any f : PV -> {T,F},
        \Sigma^f |- \tau .
```

```
    For k<=n, let PV_k := {p_i | i > k} = {p_{k+1}, ..., p_n}, so PV_0 = PV
    and PV_n = \emptyset. We show inductively that for any k<=n:
        (*)_k: for any f : PV_k -> {T,F},
                \Sigma^f |- \tau .

    We've seen (*)_0. Suppose (*)_{r-1}, 0<r<=n; we show (*)_r.
    Let f : PV_r -> {T,F}. Then we know
        {p_r} \union \Sigma^f |- \tau
    and
        {~p_r} \union \Sigma^f |- \tau.

    So by the fantasy rule,
        \Sigma^f |- <p_r =) \tau>
    and
        \Sigma^f |- <~p_r =) \tau>,
    so
        \Sigma^f |- <<p_r =) \tau> /\ <~p_r =) \tau>> .
    But then, by Lemma 2,
        \Sigma^f |- \tau .

    So (*)_n holds, i.e.
        |- \tau .
    QED

So we have proven

Theorem [soundness and completeness of PROP]:
    For any wff \tau, |= \tau iff |- \tau.

Digression:
    The strong version of completeness
        \Sigma |= \tau implies \Sigma |- \tau
    is true. For *finite* \Sigma, this follows by a similar argument to that
    for strong soundness.

    To show it for infinite \Sigma, the only difficulty is to see that if
    \Sigma |= \tau, then actually there's some _finite_ \Sigma' (= \Sigma such
    that \Sigma' |= \tau. That takes a little thought; it's equivalent to
    compactness of Cantor space 2^\omega.
```