Recall From Last Time

Boolean Syntax:

Boolean variables: $X = \{x_1, x_2, x_3, \ldots\}$

A boolean variable represents an atomic statement that may be either true or false. There may be infinitely many of these available.

Boolean expressions:

- atomic: x_i, $\top$ ("top"), $\perp$ ("bottom")
- $(\alpha \vee \beta)$, $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$, for α, β Boolean expressions

Note that any particular expression is a finite string, and thus may use only finitely many variables.

A *literal* is an atomic expression or its negation: x_i, $\neg x_i$, $\top$, $\perp$.

As you may know, the choice of operators is somewhat arbitary as long as we have a *complete set*, one that suffices to simulate all boolean functions. On HW#1 we argued that $\{\wedge, \vee, \neg\}$ is already a complete set.

A boolean expression has a meaning, a **truth value** of true or false, once we know the truth values of all the individual variables.

A **truth assignment** is a function $T : X' \subseteq X \to \{\textbf{true}, \textbf{false}\}$, where X is the set of all variables. An assignment is *appropriate* to an expression φ if it assigns a value to all variables used in φ.

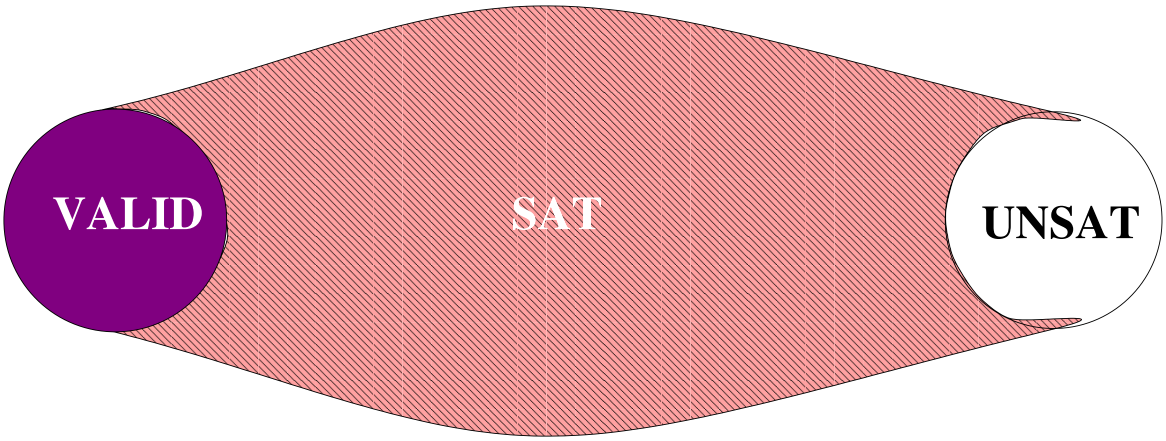
The double-turnstile symbol $\models$ (read as "models") denotes the relationship between a truth assignment and an expression. The statement "$T \models \varphi$" (read as "T models φ") simply says "φ is true under T".

If T is appropriate to φ, we *define* when $T \models \varphi$ is true by induction on the structure of φ:

- $\top$ is true and $\bot$ is false for any T,
- A variable x_i is true iff T says that it is,
- If $\varphi = (\alpha \wedge \beta)$, $T \models \varphi$ iff both $T \models \alpha$ and $T \models \beta$,
- If $\varphi = (\alpha \vee \beta)$, $T \models \varphi$ iff either $T \models \alpha$ or $T \models \beta$ or both,
- If $\varphi = (\alpha \rightarrow \beta)$, $T \models \varphi$ unless $T \models \alpha$ and $T \not\models \beta$,
- If $\varphi = (\alpha \leftrightarrow \beta)$, $T \models \varphi$ iff $T \models \alpha$ and $T \models \beta$ are both true or both false.

Definition 6.1 A boolean expression φ is *satisfiable* iff there exists $T \models \varphi$.

φ is *valid* iff for all T appropriate to φ, $T \models \varphi$. ♠



Proposition 6.2 *For any boolean expression φ,*

$$\varphi \in \mathit{UNSAT} \qquad \Leftrightarrow \qquad \neg\varphi \in \mathit{VALID}$$

$$\text{UNSAT} \;\leq\; \text{VALID}; \qquad \text{VALID} \;\leq\; \text{UNSAT}$$

Proposition 6.3 • φ *is* unsatisfiable *iff* $\varphi \;\equiv\; \bot$.

- φ *is* satisfiable *iff* $\varphi \;\not\equiv\; \bot$.
- φ *is* valid *iff* $\varphi \;\equiv\; \top$.

A Fitch proof is a sequence of expressions, each one of which is justified in terms of previous ones. There are twelve **proof rules** that tell us when a statement is justified.

Fitch has no **axioms** (statements assumed to be true without proof) but we typically start with some **premises** and reach a **conclusion** that follows from those premises.

If from a set P of premises we can derive φ, we write $P \vdash \varphi$, read as "P proves φ". This **single turnstile** symbol $\vdash$ is not to be confused with the double turnstile symbol $\models$.

These are conveniently listed on pages 557-9 of [BE].

$\wedge$ **Intro:** From $P_1, \ldots, P_n$, derive $P_1 \wedge \ldots \wedge P_n$.

$\wedge$ **Elim:** From $P_1 \wedge \ldots \wedge P_n$, derive P_i.

$\vee$ **Intro:** From P_i, derive $P_1 \vee \ldots \vee P_n$.

$\vee$ **Elim:** From $P_1 \vee \ldots \vee P_n$, if you have derived S separately from each P_i, derive S.

$\neg$ **Intro:** If you have derived $\perp$ from P, derive $\neg P$.

$\neg$ **Elim:** From $\neg\neg P$, derive P.

$\perp$ **Intro:** From P and $\neg P$, derive $\perp$.

$\perp$ **Elim:** From $\perp$, derive P (any P!)

$\rightarrow$ **Intro:** If you have derived Q from P, derive $P \rightarrow Q$.

$\rightarrow$ **Elim:** From $P \rightarrow Q$ and P, derive Q (also called **modus ponens**).

$\leftrightarrow$ **Intro:** If you have derived Q from P and have derived P from Q, derive $P \leftrightarrow Q$.

$\leftrightarrow$ **Elim:** From $P \leftrightarrow Q$ and P, derive Q.

Now that we've defined the propositional subset of Fitch, $\mathcal{F}_T$, we consider some important properties of a proof system:

Soundness: If a statement φ can be proved from a set of statements T, then φ is a tautological consequence of T. (If $T \vdash \varphi$, then $T \models \varphi$.)

Completeness: If φ is a tautological consequence of T, then φ can be proved from T. (If $T \models \varphi$, then $T \vdash \varphi$.)

Compactness: If every finite subset of T can be satisified by some assignment, then there exists an assignment satisfying all of T.

The intuition behind soundness is clear: each of the rules corresponds correctly to the meaning of the symbols, so none of them should be able to prove false things from true premises. Our proof will follow this intuition.

Let's restate compactness in a form that's easier to prove by induction:

Theorem 6.4 *Let φ be a statement in a proof P and let $\alpha_1, \ldots, \alpha_n$ be the premises in force when φ occurs. Then any truth assignment that makes each α_i true also makes φ true.*

Proof: [BE] says "if the conclusion ever fails for any step in any proof, consider the first step on which it fails". I would be more inclined to view proofs as being inductively constructed by adding steps. In either case, it suffices to prove the conclusion in the case in which every statement of P *before* φ is a tautological consequence of its premises.

For φ to occur in the proof, it must have been produced by some rule. We thus are reduced to twelve cases, one for each of the rules. We'll do a few of these, and do a few more in the exercises.

$\wedge$ Intro:

φ is of the form $\beta_1 \wedge \ldots \wedge \beta_k$ for statements β_i in its scope. Each of the β_i is a tautological consequence of its premises by the (inductive) hypothesis, and each of those premises is in force for φ by the scoping rules.

So any truth assignment that makes all of φ's premises true must make all the β_i's true as well. And then φ must be true, by the definition of truth for statements involving $\wedge$.

$\rightarrow$ Elim:

There are statements of the form β and $\beta \rightarrow \varphi$ in the scope of φ. As above, each of these two statements is true under any truth assignment that makes all the premises of φ true. By the definition of truth for statements involving $\rightarrow$, it is not possible for β and $\beta \rightarrow \varphi$ to be true when φ is false.

$\to$ Intro:

Here φ is of the form $\beta \to \gamma$ and there is a subproof, within the scope of φ, of γ from β. The premises in force during that subproof were a subset of the premises in force for φ.

So in any truth assignment that makes all of φ's premises true, the premises for the subproof are also true. Since the inductive hypothesis holds for γ, any assignment that *also* makes β true must make γ true.

Thus any such assignment *either* makes β false or makes γ true, and by the definition of truth for $\to$ either of these makes $\beta \to \gamma$, and this statement *is* φ.

$\perp$ Elim:

Here φ is an arbitrary sentence, and the statement $\perp$ is in its scope. Since the inductive hypothesis holds for $\perp$, we know that any assignment making the premises of $\perp$ true makes $\perp$ *itself* true. There can thus be no such assignment. But this means no assignment could make all the premises of φ true, since the premises of $\perp$ are a subset of these. The conclusion for φ is thus vacuously satisfied.

Remaining Steps: There are eight of these, with proofs similar to the above four cases. You'll be assigned to write out some of these on HW#2.

♠

This proof is emotionally unsatisfying in the way that many proofs in mathematical logic are unsatisfying. We seem to be proving that a step is valid if and only if it's valid, and not really saying anything at all.

The key point are that *truth* and *provability* are two different properties, even though for sound and complete proof systems they hold for *exactly the same sentences*. The *reason* that this system is sound and complete is that the proof steps correspond properly to the definitions of truth for each operator, and vice versa. We shouldn't be confident that this correspondence holds until we've checked it in each case.

We've just shown that "everything provable is true" for propositional Fitch. The next step is to show that "everything true is provable". The natural thing to do would be to *construct* a Fitch proof for an arbitrary tautological consequence. Unfortunately, we don't have a useful inductive definition for tautological consequences the way we did for proofs.

What we'll do instead is to prove that if φ is *not* provable from T, then there is *some* truth assignment making T true and φ false.

Lemma 6.5 *φ is provable from T iff $\perp$ is not provable from $T \cup \{\neg\varphi\}$.*

A set of sentences from which you *can't* prove $\perp$ is called **formally consistent**. A set of statements that has some truth assignment making them all true is called **tt-satisfiable**. The completeness result will follow once we show that every formally consistent set of sentences is tt-satisfiable.

Our proof will require one more definition. A set of sentences T is called **formally complete** if for *every* sentence φ, either $T \vdash \varphi$ or $T \vdash \neg\varphi$. (Note that we are not saying that "formal completeness" has anything to do with "completeness" – one is a property of sets of statements and the other a property of the whole proof system.)

We will show:

- Every formally consistent, formally complete set of sentences is tt-satisfiable.
- Every formally consistent set of sentences can be extended to a formally complete set while remaining formally consistent.

Since a truth assignment satisfying an extension of T also satisfies T itself, these two statements imply that any formally consistent set is tt-satisfiable, and thus imply the completeness of propositional Fitch.

Let's first show that a formally consistent, formally complete set of sentences is satisfiable. Suppose that T is such a set. Let x be any of the *variables* of the system. Since "x" is a sentence, we know that either $T \vdash x$ or $T \vdash \neg x$. Could both be true? No, because then by writing both these proofs and doing one $\perp$-Intro step we could prove $\perp$ from T.

This tells us how to define our truth assignment! We set each x_i to be true if $T \vdash x_i$ and false otherwise, giving us a mapping h from the set of variables to the set of truth values. No truth assignment *other than* h could make all of T true, but we still have to show that h itself *does* makes T true.

What we can do is to prove that for *any sentence* φ, $T \vdash \varphi$ iff this truth assignment h makes φ true. This requires induction on the structure of sentences:

- Any T can prove $\top$, and no formally consistent T can prove $\bot$.
- If φ is x_i or $\neg x_i$ it follows from the definition of the assignment h that φ is true iff T proves it.
- If φ is $\alpha \wedge \beta$, and α and β are each provable from T iff they are true, we show that φ is provable from T iff both α and β are. If φ is provable we can prove either α or β in one more step using $\wedge$-Elim. If we have proofs of both α and β, we write them both down and then prove φ in one more step by $\wedge$-Intro.
- If φ is $\neg\alpha$, and α is provable iff it is true under h, we must show that φ is. This just follows from the definitions, as formal completeness says we must be able to prove either α or $\neg\alpha$, and formal consistency says we cannot prove both.
- The cases for $\vee$, $\rightarrow$, and $\leftrightarrow$ are similar to that for $\wedge$, using the appropriate proof rules.

So we have our result for formally complete and formally consistent sets. Now we must show that if T is formally consistent we can extend it (by adding sentences to it) to make it formally complete while keeping it formally consistent.

We consider the atomic variables x_i in order, a potentially infinite process. At stage i of our process we look at x_i and $\neg x_i$ and ask whether either is provable from T as extended so far. If one of them is already provable we do nothing. But if *neither* is provable, we add x_i (an arbitrary choice) to T and go on.

At the end of this process, T has become formally complete, because we just proved that a system that proves each variable or its negation proves each sentence or its negation. Could it fail to be formally consistent?

The only way we could have destroyed the formal consistency property would be when we added a variable x_i to T. But we only did this if the former T could not prove $\neg x_i$. If there *were* a proof of $\perp$ from T and x_i, we could adapt this proof to prove $\neg x_i$ from T alone by $\perp$-Elim – just take x_i as the premise in the presence of T and derive $\perp$.

We are done! The "infinite process" might worry you a bit, and perhaps it should. We can't carry out this process with any kind of finite algorithm. But all we want to argue is that the final T *exists*, and it does – for any individual x_i, it is well-defined whether it is in the final T or not.

To review one more time:

- If T is formally consistent we extend it to be formally complete, and
- we then know that it settles all sentences consistently with the assignment it places on the variables,
- so the original T is tt-satisfiable, by this assignment.

You've noticed that we frequently want to talk about *infinite* sets of sentences T, such as the set of all valid sentences about variables $x_1, x_2, \ldots$ or the set of all satisfiable sentences. Each individual statement is itself finite, and each proof is finite, but we might worry whether a property such as consistency is somehow different for finite and infinite sets.

Theorem 6.6 *(***Compactness of Propositional Logic***) Let T be a set of propositional sentences. If every finite subset of T is tt-satisfiable, then T itself is tt-satisfiable.*

Proof: Suppose that T is not tt-satisfiable. By the Completeness Property of $\mathcal{F}_T$, there exists a proof of $\perp$ from T. Since this proof is finite, it uses only a finite subset S of the sentences of T as its premises. But then S itself proves $\perp$ and by the Soundness Property it cannot be tt-satisfiable. We have proved the contrapositive of the theorem and thus have proved the theorem. ♠

Note that compactness is a *semantic* rather than a *proof-theoretic* property. Its definition for propositional logic depended on tt-satisfiability, which depends on the notion of a truth assignment being a model for a set of sentences.

More generally, a **semantics** for a logical system is a way of defining models and whether a given model satisfies a given set of sentences. The general **compactness property** of such a system can be stated: "If every finite subset of T has a model, then T has a model."

Our proof of compactness for propositional logic used no *specific* properties of propositional logic, only the fact that it has a proof system that is sound and complete, and in which every proof is finite.

Propositional Fitch is not the only sound and complete proof system for propositional logic. [BE] introduces another system called **resolution** in section 17.4, and there are many others.

The **relative power** of such proof systems is an active area of study in computer science and mathematical logic. *Qualitatively*, of course, any two sound and complete systems have the same power, because they can each prove the true tautological consequences and no others.

But *quantitatively* there are differences among the systems, some proved and some conjectured. For example, there are families of tautologies that have polynomial-size proofs in propositional Fitch but require *exponential-size* resolution proofs. We won't prove this in this course, but we'll later talk about how such results relate to computational complexity.

Resolution is best understood as a **refutation system**, a way of showing that a particular set of sentences is *not* tt-satsifiable. Express each sentence in CNF, so that it is a set of clauses each of which must be satisfied. Of course an AND of CNF sentences becomes a single CNF sentence.

Resolution proceeds by adding new clauses called **resolvents** to the set of clauses, in such a way that the tt-satisfiability of the set remains the same. The basic rule is to take a clause containing a literal b, and another clause containing the literal $\neg b$, and form the resolvent by unioning the clauses together and removing both b and $\neg b$. It is not hard to see that to satisfy both the original clauses you must satisfiy the new one.

The process ends when the **empty clause** appears, meaning that the set of clauses is no longer satisfiable, or when further taking of resolvants does not increase the set, meaning that it is.

This process can be automated and is thus sometimes a practical theorem-prover for medium-sized problems. [BE] says that something like it is used to implement the TautCon instruction in Fitch.