Set Theory and Logic

February 4, 2009

Computable Functions

Notation: $\lambda x.p(x)$, where p is an expression, denotes that x is a bound variable.

Definition: A (computable) primitive recursive function $\mathbb{N}^k \to \mathbb{N}$ is defined recursively by: $S : x \mapsto x + 1$ is computable, $\lambda x.0$ is computiable, $\text{proj}_m^n(x1, \ldots, x_n) = x_m$ is computable, [we may compose functions nonrecursively: if $g_1, \ldots, g_k$ are primitive recursive then so is $f(g_1, \ldots, g_k)$ - I believe this was defined incorrectly in lectures], and if g, h are computable and f is defined by $f(x_1, \ldots, x_k, 0) = g(x_1, \ldots, x_k)$, $f(x_1, \ldots, x_k, S(n)) = h(x_1, \ldots, x_k, n, f(x_1, \ldots, x_k, n))$ then f is also computable. This is called primitive recursion (note we may only recurse on the last variable). The collection of functions so defined are the primitive recursive functions. It is not obvious that e.g. the Fibonacci number function $\text{Fib}(n)$ is primitive recursive; however, it is the case that the pairing and unpairing functions $p(x, y), \pi_1(x), \pi_2(x)$ are primitive recursive, so writing $\langle x, y \rangle$ for an ordered pair and $1\text{st}(x), 2\text{nd}(x)$ for $\pi_1(x), \pi_2(x)$ we can define $F(0) = \langle 1, 1 \rangle, F(n+1) = \langle 2\text{nd}(F(n)), 1\text{st}(F(n)) + 2\text{nd}(F(n)) \rangle, \text{Fib}(n) = 2\text{nd}(F(n))$.

Note: These are all total functions, by structural induction.

Definition: The Ackermann function $A(x, y)$ is defined by $A(x+1, y+1) = A(x, A(x+1, y)), A(x, 0) = x + 1, A(0, y) = 1$. This is obviously computable, but not primitive recursive. As a proof sketch, this is because it dominates every primitive recursive function. Definition: For $f, g : \mathbb{N} \to \mathbb{N}$, f dominates g if $\exists n_0 \in \mathbb{N}$ such that $\forall n > n_0, f(n) > g(n)$. We can also define domination for multivariate functions. We prove $A(x, y)$ dominates f, for all primitive recursive f, by structural induction.

Lemma: $A(x, y)$ is computable, by induction on $\mathbb{N} \times \mathbb{N}$ with the lexicographic ordering: let $\langle x+1, y+1 \rangle$ be the least pair such that $A(x+1, y+1)$ is undefined, then $A(x+1, y)$ is defined, and so $A(x, A(x+1, y))$ is defined.

The Ackermann function gives us a hint of how to construct n-recursive functions which are not $(n-1)$-recursive. There are even functions which are (computable) total but not n-recursive for any n, e.g. Goodstein: Consider taking a number n (e.g. 7 say) and expanding it in powers of two, then expanding their exponents in powers of two and so on, e.g. $2^2 + 2 + 1$ is this case. Then replace 2 with 3 and subtract 1: $3^3 + 3$; replace 3 with 4 and subtract 1: $4^4 + 1 + 1 + 1$, and so on. The Goodstein function of n is defined as the number of times we can do this before reaching 0.

This tells us recursion isn't the "correct" notion of computable functions. Instead, we have to accept that computable functions need not be total, e.g. $\lambda n.\ \bot$, the everywhere undefined function, is trivially computable by a program which inputs n and then loops forever. Hand-in-hand with this comes the notion

of inversion or minimization: for $f : \mathbb{N} \to \mathbb{N}$ computable, $f^{-1}(x)$ is the least n such that $f(n) = x$. If f is total computable then f^{-1} is computable (but not necessarily total). (We cannot form f^{-1} for a general computable f; given such a function we can compute "an inverse" for a given n by some sort of "diagonal strategy", e.g. performing the first step of computing $f(1)$, then the first step of computing $f(2)$, the second step of computing $f(1)$, the first step of computing $f(3)$, the second step of computing $f(2)$, the third step of computing $f(1)$ and so on, and stopping if we finish computing $f(x) = n$ for some x. But this inverse x thereby found is in no sense canonical; it is extremely sensitive to the precise computing strategy we used).

Therefore, we define: the General Recursive Functions, also called Partial Recursive Functions or General Computable Functions, have the same definition as that of primitive recursive functions with the addition of inversion, but subject to the constraint that "you are only allowed to do it once" [the lecturer appeared to be saying that this can replace primitive recursion in the definition, but that is frankly ludicrous]. This is "correct": we will later prove that any "legitimate" function definition, i.e. any definition constructed by using all the permitted techniques arbitrarily many times but only applying inversion to functions which are in fact total, is equivalent to one in the required form.

What we have been doing so far is a "syntactic" characterisation of computability. There is also a "semantic" characterisation in terms of machines. Two architectures appear in the literature; Turing machines and Minsky (aka Register) Machines. (A Turing machine is a Finite State Machine (i.e. a set of states and a transition function from pairs (state,input) to states; one state is usually identified as the "start" state and some states "final" or "accepting" states, where a machine in such a state at the completion of input is considered as giving a positive result) which takes inputs froma tap and may move the tape one space forward or backward and/or write on the tape as a "side effect". A register machine uses (mathematically idealised) registers which hold members of $\mathbb{N}$ and insturctions. Either type of machine can simulate the other). These two capture a notion of computation which is "finite but unbounded" (e.g. a Turing machine's tape is infinite in both directions, but the input program is finite), and deterministic.

It is eminently plausible (if too fiddly for the lecturer to be capable of giving a proof) that any syntacticly expressible function can be computed by such a machine. In fact the two notions of computability are equivalent. To see this, consider that since the machines are deterministic, we can form the function $T(m, i, t)$, which outputs a "cine film" of the first t steps undertaken by a Turing machine whose description is encoded by m when given the input i. This $T : \mathbb{N}^3 \to \mathbb{N}$, Kleene's T-function, is primitive recursive, though this fact is nonobvious.

If a machine M exists to compute a function f - i.e., M, when input with n, proceeds through finitely many steps and eventually reaches some "halt" state, at which point we can observe the value $f(n)$ from the machine in some fashion - then we can calculate this using T - we seek the least t such that $T(m, i, t)$ has, as its last element, a description which says M has halted, and then extract the "magic number" from the last "frame" of our "film".

So, this notion exists both syntacticly and semanticly; therefore it is in some sense "natural" and "important". However, is it the "correct" notion of computability? In some sense this is a mathematical question which can

never be proven - for, if one had a proof that this was so, one would need a characterisation of "the correct notion of computability", which would mean that the statement was false. The Church-Turing thesis is the assertion that this is the correct notion of computability.

Some evidence in favour of this comes from the fact that there is another characterisation of the same thing: the λ-calculus. We write e.g. $(fx)x$ to mean $(f(x))(x)$; we have expressions like $\lambda x.[]$, $\lambda x.x$, called I, $\lambda x.\lambda y.x$, called K, $\lambda f.\lambda x f(fx)$, known as 2, $\lambda f.\lambda x f(f(fx))$ which is called 3 and so on; these are the Church numerals. $\lambda n \lambda f \lambda x.f(nfx)$ is the successor function, yielding the successor of n; $\lambda n \lambda m.\lambda f \lambda x (nf)(mfx)$ is plus. Note that since our functions are only of one variable, rather than being an $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ function this is a $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ function. A little consideration will see that any function can be expressed naturally in this form; this is known as Currying. $\lambda n \lambda m \lambda f.\lambda x n(mf)x$ is multiplication; as an exercise the reaxer should consider the function $\lambda n \lambda m \lambda f n m f$.

There are notions of e.g. true and false in this formalism, which enable us to construct many things.

Consider e.g. fact n = if n=0 then 1 else n * fact(n-1). This is not an obviously valid definition; we need to prove by induction that it is well defined. Consider $F : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$: $Ff = \lambda n$ if $n = 0$ then 1 else $n \times f(n-1)$. Suppose that F has a fixed point f, i.e. $Ff = f$, i.e. $\forall n, f(n) = Ffn =$ if $n = 0$ then 1 else $n \times f(n-1)$. So if such a fixed point exists, we have a factorial function - more generally, if such fixed points always exist, then we may define functions by recursion in this fashion.

Define: $Y = \lambda f.(\lambda x f(xx))(\lambda x f(xx))$. Observe that e.g. $Yg = (\lambda x.g(xx))(\lambda x.g(xx)) = g((\lambda x.g(xx))(\lambda x.g(xx))) = g(Yg)$; thus Yg is a fixed point of g.

We say a function f is λ-computable if there is a λ-term which, when applied to the Church numeral n, gives the Church numeral $f(n)$. The lecturer refused to be drawn on the decidability of this proposition. In any case, this turns out to give the same characterisation of computability as that above.

Decidable sets of natural numbers

Our first throught: $X \subset \mathbb{N}$ is <u>decidable</u> (or in old notation, recursive) iff there is a total computable $f : \mathbb{N} \to \{0, 1\}$ such that $f^{-1\prime\prime}\{1\} = X$ (this is notation: in normal mathematics, it is not necessary to distinguish between a function being applied to a value $f(x) = y$, or applied to a set $f(X) = \{f(x) : x \in X\}$, as we can tell these from context. But in set theory the distinction between sets and values is more fluid, so we need the notation: we use $f'x$ for the first case and $f''X$ for the second).

A notion which turns out to be more useful in practice: $X \subset \mathbb{N}$ is semidecidable (recursively enumerable) iff there is a partial recursive $f : \mathbb{N} \to \{0, 1\}$ with $f^{-1\prime\prime}\{1\} = X, f''X = \{1\}$ - "members of X can be recognised", "membership in X can be confirmed in finite time".

There are several ways to define such a set: we might want to define a semidecidable set to be $f''\mathbb{N}$ of a partial computable function $\mathbb{N} \to \mathbb{N}$, or the same for a total computable f, or even $\{n : f(n) \downarrow\}$ for some parallel computable $f : \mathbb{N} \to \mathbb{N}$; fortunately these notations are all equivalent (Notation: $\downarrow$ denotes "halts", $\uparrow$ "does not halt"), as can be proved reasonably easily. E.g. given $f, f''\mathbb{N} = X$ and f partial computable, we can calculate a total computable g with $g''\mathbb{N} = X$ by diagonal computation: $g(n)$ computes the first step of $f(1)$,

then the first step of $f(2)$, the first two steps of $f(1)$, the first step of $f(3)$, the first two steps of $f(2)$ and so on, and outputs the nth value output from one of these. (We might think it was better to "store" the state and "resume" to perform one more step in each time, but in fact the proof is easier if we simply recalculate and do one more step each time). (As will be common in the course, we are ignoring the finite case; it would be easy to find a suitable g if X is merely a finite set). Similarly, for a parallel computable h which halts only on those values which are $\in X$, run f on all values in parallel, and if f outputs n, halt (outputting, say, 0). The remaining cases are left as an exercise.

Write $e\{n\}$ or e_n for the machine identified by n; we sometimes call n its gnumber (Gödel number).

Halting problem: can we determine in general whether $M(n)\downarrow$? Suppose we had a machine M taking ordered pairs such that $M\langle x,y\rangle\downarrow$=yes if $e_x(y)\downarrow$, $\downarrow$=no if $e_x(y)\uparrow$. Then we can construct a machine $M^\star$ with $M^\star\langle x,y\rangle\uparrow$ if $e_x(y)\downarrow$, $\downarrow$=no if $e_x(y)\uparrow$. Combine this with the machine $a\mapsto\langle a,a\rangle$ to get a machine $M^\dagger$ with $M^\dagger a\ (=M^\star\langle a,a\rangle)\uparrow$ if $e_a(a)\downarrow$, $\downarrow$ if $e_a,a)\uparrow$. Now $M^\dagger$ has a gnumber m, so $M^\dagger m=\uparrow$ if $e_m(m)\downarrow$, $\downarrow$ if $e_m(m)\uparrow$, but $e_m=M^\dagger$, so this is a contradiction, and the original machine M cannot exist. Thus, there are semidecidable sets which are not decidable, e.g. $\{\langle x,y\rangle : e_x(y)\downarrow\}=:K$, the halting set. This is the "nastiest set you need to know about" unless you study degree theory, which is the study of computation relative to an oracle, classifying "levels of nastiness". Note that the above argument holds given any oracle, so there is no maximum level of nastiness.

Important fact: There is a universal machine. Recall $T(m,i,t)$ was primitive recursive; a machine that computes T can be used to simulate any machine.

We know we can encode strings of mathematical symbols as numbers, so we can encode a theory as a set of natural numbers. Is this set decidable? An undecidable axiom set would be somewhat useless, but how about a semidecidable set? If a theory has a semidecidable set of axioms it is said to be recursively axiomatizable. Without proof, any such theory has a semidecidable set of theorems - we can form a machine to diagonally compute all axioms of the theory, and couple this to a machine which will do a "brute force" "breadth first search", generating all provable statements in the theory.

Fact(Craig): Every semidecidable theory T (i.e. one whose theorems are a semidecidable set) has a decidable axiomatization. For this we need the following aside: suppose X is semidecidable but not decidable. We have $X=f''\mathbb{N}$ where f is total computable. This f cannot emit X in increasing order (or anything remotely resembling it), as that would make X decidable: if you run f until some number larger than n is emitted, we then know all numbers $<n$ not yet emitted do not lie in X. Take our machine and modify it to ignore theorems of the form $\phi\wedge 1=1$. Then, modify again so that the nth theorem emitted has a chain of n $1=1$s appeneded to the end, $\phi\wedge 1=1\wedge 1=1\ldots$. I.e., we encode (ϕ,n) where n is the "time" when ϕ was emitted. But then the set output by this is decidable.

Given $\phi(a,b)$ computable, fix a and consider $\lambda b.\phi(a,b)$. There is a computable function which does this (using gnumbers); call it $S:(\phi,a)\mapsto\lambda b.\phi(a,b)$; and so on for functions with multiple arguments, $\boldsymbol{S}:(\phi,\boldsymbol{a})\mapsto\lambda\boldsymbol{b}.\phi(\boldsymbol{a},\boldsymbol{b})$. This is called the $S-m-n$ theorem. Changing notation again (thanks, lecturer) and writing ϕ_e for the function with gnumber e, $\phi_e(a,b)=\phi_{S(e,a)}(b)$.

Corollary (Fixed Point Theorem): Let $h:\mathbb{N}\to\mathbb{N}$ be total computable. Then

$\exists n$ such that $\phi_n = \phi_{h(n)}$: consider the map $(e, x) \mapsto \phi_{h(S(e,e))}(x)$. This is computable, say its computed by the ath machine. Then set $n := S(a, a)$, then $\phi_n(x) = \phi_{S(a,a)}(x) = \phi_a(a, x) = \phi_{h(S(a,a))}(x) = \phi_{h(n)}(x)$. This tells us that there are many pairs of distinct machines which compute the same function.

The halting set is an example of a general phenomenon: Intension vs Extension. The terminology is from medieval philosophy, but useful to us: "intention" is the "meaning" of a function - in our case, the syntax of its declaration - while "extension" is its "meaning" or "extent" - the "graph" of the function.

Rice's theorem: Let A be a nonempty proper subset of the set of all computable functions in one variable. Then $\{n : \phi_n \in A\}$ is not decideable: find natural numbers a, b such that $\phi_a \in A, \phi_b \in A$. Aside: $\chi(A)$ generally denotes a characteristic function of A in the sense λn. if $n \in A$ then 1 else fail (possibly not halting). Assume A is decidable so we can form $\chi(A)$ to be total, i.e. it gives 1 on elements of A, 0 on nonelements of A. So we can define $g(n) =$ if $\phi_n \in A$ then b else a computable. By the fixed point theorem there is n such that $\phi_n = \phi_{g(n)}$. If $\phi_n \in A$ then $\phi_{g(n)} = \phi_n \in A$ and $g(n) = b$ by construction of g [so $\phi_{g)(n)} \notin A$]. But if $\phi_n \notin A$ then $\phi_{g(n)} = \phi_n \notin A$ but $g(n) = a$. Thus we have a contradiction.

Natural deduction for propositional logic: Any binary connector is characterised by its truth table. We can construct all possible ones out of some basic ones; it's possible to construct all from | (NAND), but more practical to use $\wedge, \vee \rightarrow$; also $\neg$, which is shorthand: $\neg A = A \rightarrow \bot$. $\bot$ is always false. There are several basic rules: the elimination (or, in nonstandard terminology, "use") rules for $\wedge$: $\frac{A \wedge B}{A}$, $\frac{A \wedge B}{B}$, for $\rightarrow$ $\frac{A, A \rightarrow B}{B}$, and for $\bot$, $\frac{\bot}{A}$, and the introduction rules for $\wedge$, $\frac{A, B}{A \wedge B}$ and $\vee$, $\frac{B}{A \vee B}$, $\frac{A}{A \vee B}$ (we shouldn't really need two cases here; this is a limitation of our linear proof notation). There are also two more complicated rules: $\rightarrow$ introduction, $\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \rightarrow B}$ (the dots denoting a proof of B from A, and the square brackets denoting that we have "used up" the A and no longer need to suppose it for the resulting statement to be true). And finally or elimination: $\dfrac{A \vee B \quad \begin{array}{cc}[A] & [B] \\ \vdots & \vdots \\ C & C\end{array}}{C}$, somewhat the "hardest" of the rules. Aside: it is natural to generalize or elimination to sets of n variables, $A \vee B \vee C \vee D$; in this case $\frac{\bot}{A}$ is simply the empty case of this.

We can use these rules to find "natural" proofs for various tautologies; work "backwards" using the elimination rules, to obtain our premises, and then "forwards" to try and obtain the conclusions we need, e.g. to proove $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \wedge B) \rightarrow C$, we first use $\rightarrow$ elimination to show we need to prove $(A \wedge B) \rightarrow C$ from $A \rightarrow (B \rightarrow C)$, and then again to find we need to prove C from that and $A \wedge B$. It is then relatively easy to complete the proof (obtaining A from $A \wedge B$ and using this with $A \rightarrow (B \rightarrow C)$ to get $B \rightarrow C$, then using $A \wedge B$ again to get B and combining these to get C.

To be able to prove all tautologies we also need classical negation: $\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}$,

or alternatively $\frac{\neg\neg A}{A}$.

The "obvious" proof strategy described above does not always work; e.g. to prove $A \wedge (B \vee C) \to (A \wedge B) \vee (A \wedge C)$, we need to avoid expanding fully immediately; we should use $\to$ introduction, but not or introduction until later in the proof. Thus proving in this system doesn't seem like something that can obviously be done in polynomial time [whether it can is an open question, if I understand the lecturer correctly].

Disjunctive normal forms: this gives a standard representation for any formula. Given e.g. $F(A, B, C)$, simply form the truth table of F, and then write out the disjunction of the combinations on which F is true - e.g. if F is true when A is false, B is true and C is false, and also true when A is true, B is true and C is true, we could write it as $(\neg A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C) \vee \ldots$. There is also a Conjunctive Normal Form, but this is far harder to understand intuitively. We can use these normal forms to prove the completeness theorem; we then only need to show that we can prove any true statement in disjunctive normal form, which is far eaiser to do inductively than showing we can prove a general true statement.

Consider forming a proof step-bp-step; if we want to prove $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$, at first we see our proof will take the form $\begin{array}{c}A \to (B \to C)\\ \vdots\\ (A \to B) \to (A \to C)\end{array}$, then it becomes clear it will be $\begin{array}{ccc}A \to (B \to C) & & \\ \vdots & & A \to B\\ \vdots & & \vdots\\ A \to C & & \end{array}$,

etc. But we don't know "how much space to leave" in the areas denoted by the dots, which can be an obstruction to doing this mechanically.

Thus, some prefer to reason about sequents. $\vdash$ is a "metta" statement; it claims the RHS is provable from the LHS. So we would consider our first statement as $\vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C))$, and then we know that it follows from $A \to (B \to C) \vdash (A \to B) \to (A \to C)$, which in turn will follow from $A \to (B \to C), A \to B \vdash A \to C$. These statements with $\vdash$ are called sequents. The rules for $\vdash$ are: for $\wedge$, the rule on the left is $\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C}$, and the rule on the right is $\frac{\Gamma \vdash A, \Gamma \vdash B}{\Gamma \vdash A \wedge B}$; for $\vee$, on the left $\frac{\Gamma, A \vdash C, \ldots, \Gamma, B \vdash C}{\Gamma, A \vee \cdots \vee B \vdash C}$ and on the right $\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$; for $\to$, $\frac{\Gamma, vdash A, \Gamma, B \vdash C}{\Gamma, A \to B \vdash C}$ and $\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$. Note that here we always only have one formula on the right. This is actually insufficient to give us all of classical logic; we only obtain "constructive knowledge". To obtain full classical logic we need to relax this constraint and allow many formulae on the right; we now consider $\Gamma \vdash \Delta$ to mean: if everything in Γ is true then something in Δ is true. The additional rules are: on the left, $\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}$ and on the right $\frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}$.

Natural deduction rules for the predicate calculus: there are two simple rules, $\forall$-elimination $\frac{\forall x F(x)}{F(a)}$ and $\exists$-introduction $\frac{F(a)}{\exists x F(x)}$, and two more compli-

cated ones: $\exists$-elimination, $\dfrac{\exists x F(x) \qquad \begin{matrix}[F(a)] \\ \cdots \\ P\end{matrix}}{P}$ where "a" must not apear free in the premises of "P", and $\forall$-introduction, also called universal generalization: $\dfrac{\begin{matrix}\vdots \\ F(a)\end{matrix}}{\forall x F(x)}$, where "$a$" must not be free in the assumptions. The equivalent sequent rules are on the left $\frac{\Gamma, F(a) \vdash \Delta}{\Gamma, \forall x F(x) \vdash \Delta}$ and on the right $\frac{\Gamma \vdash \Delta, F(a)}{\Gamma \vdash \Delta, \forall x F(x)}$ with the "side conditions" that "a" must not appear free in the lower sequent, on the left $\frac{\Gamma, F(a) \vdash \Delta}{\Gamma, \exists x F(x) \vdash \Delta}$ with, if you trust the lecturer, the same side conditions, and on the right $\frac{\Gamma \vdash \Delta, F(a)}{\Gamma \vdash \Delta, \exists x F(x)}$.

Aside: we call the formula we are "using" from our assumptions the eigenformula, and the one in the target the eigenvariable

There is one last rule, which is neither really on the left nor on the right: $\dfrac{\Gamma \vdash \Delta, A \qquad A, \Gamma' \vdash \Delta'}{\Gamma \cup \Gamma' \vdash \Delta \cup \Delta'}$. This makes finding proofs much harder, since it lacks the subformula property; for all other rules, everything appearing on the top is a subformula of the bottom. So our "search space" suddenly becomes infinite.

We seek reassurence that any sequent provable by means of cut is also provable without it. Might there be a method that takes a proof of a sequent and processes it into a proof without cut - "cut elimination"?

Note that "$\vdash$" - the sequent that true implies false - must not be provable for the system to be consistent. Since by cut, $\dfrac{\vdash A \qquad A \vdash}{\vdash}$, a proof of cut elimination is equivalent to a proof of consistency for the system - thus it must not be easy.

Consider a proof like: $\dfrac{\dfrac{\Gamma \vdash \Delta, A \qquad \Gamma, A, B \vdash \Delta}{\Gamma \vdash A, B} \qquad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}$.

Note that while we cut again below, the things we're cutting on are shorter; therefore, we look to find a well-ordering of proofs, and then consider shorter things as "more important".

ϵ-terms: Consider $\vdash (\exists x)(\forall y)(F(y) \to F(x))$. This isn't provable constructively. The only way to get this is $\exists$ on the right, but $(\forall y)(F(y) \to F(x))$ is silly. Now, a trick: keep a copy of the formula on the right. $\dfrac{\vdash (\forall y)(F(y) \to F(x))}{\vdash (\exists x)(\forall y)(F(y) \to F(x))}$ $\vdash ($

by $\exists$-R. Note: we can contract $\frac{\Gamma \vdash \Delta, \phi, \phi}{\Gamma \vdash \Delta, \phi}$. Continuing, $\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F(b), F(c) \vdash F(a), F(b)}{F(b) \vdash F(a), F(c) \to F(b)}}{\vdash F(b) \to F(a), F(c) \to F(b)}}{\vdash F(b) \to F(a), (\forall y)(F(y) \to F(b))}}{\vdash F(b) \to F(a), (\exists x)(\forall y)(F(y) \to F(x))}}{\vdash (\forall y)(F(y) \to F(a)), (\exists x)(\forall y)(F(y) \to F(x))}}{\vdash (\exists x)(\forall y)(F(y) \to F(x))}$.

Essentially, this is the law of excluded middle, or proof by cases (either $F(y)$ is always false, so $F(y) \to F(x)$ is always true, or $F(y)$ is true for some $y = y_0$ which suffices as a witness).

This is useful because it lets us introduce constants with axioms: $(\epsilon x)(F(x))$ (this is an ϵ-term).

Proposition: Every theory in a countable language can be extended to a complete theory: Let $\phi_0, \phi_1, \ldots$ enumerate the closed formulae of $\mathcal{L}$. Let $T_0 = T$; set $T_{n+1} = T_n \cup \{\phi_n\}$ if this is consistent, T_n otherwise. Then $T_\omega = \bigcup_{n<\omega} T_n$ is complete.

Theorem: Every consistent theory in first-order logic has a model: Let T be a consistent theory, let $T^\star$ be complete. Enlarge $\mathcal{L}(T)$ by adding ϵ-terms for every ϕ such that $T \vdash (\exists x)(\phi(x))$ (add in $(\epsilon x)(\phi(x))$). Call this languag e$\mathcal{L}_1$. In this new language, $T^\star$ is no longer complete, so complete it to $T_1^\star$. Iterate ω times and take the union $T_\infty^\star$; any model for $T_\infty^\star$ models T as well. Then obtain a model for $T_\infty^\star$ by taking all the ϵ-terms we have constructed en route; the reader should verify that this is indeed a model.

There is "no entity without identity" - entities are only interesting if we have a robust notion of when two of them are the same. Are proofs valid entities? In particular, does removing cuts bring proofs to a normal form? No - consider $\frac{\Gamma\vdash\Delta}{\Gamma\vdash\Delta,A}$ - "weakening-R". Consider $\dfrac{\dfrac{\begin{array}{c}D_1\\ \vdots\\ \Gamma\vdash\Delta\end{array}}{\Gamma\vdash\Delta,A} \qquad \dfrac{\begin{array}{c}D_2\\ \vdots\\ \Gamma\vdash\Delta\end{array}}{A,\Gamma\vdash\Delta}}{\Gamma\vdash\Delta}$. If eliminiating cuts is ok, this tells us that D_1, D_2 are the same proof - i.e. any two proofs of the same sequence are the same. This cannot be right. This is a reason to only allow one thing on the right (constructivism).

Definition: A filter in a boolean algebra $\mathcal{B}$ (a distributive complemented lattice) is an upward-closed subset of $\mathcal{B}$ closed under binary intersection, i.e. $F \subset \mathcal{B}$ such that $(\forall y)(y \geq x, x \in F \Rightarrow y \in F)$ and $(\forall x, y \in F)(X \wedge Y \in F)$. E.g. 1) a principal filter $\{y : y \geq a\}$ 2) cofinite subsets of $\mathbb{N} \subset \mathcal{P}(\mathbb{N})$.

Fix a Boolean algebra $\mathcal{B}$ and consider $\{F \subset \mathcal{B} : F \text{ is a filter}\}$ partially ordered by inclusion. We claim this poset is chain-complete, i.e. the union of any chain of filters is another filter. (Snotty logician-type point: this is obvious from the syntax, since filters are defined by elementary properties, so their unions must be filters). By Zorn's lemma, there is a maximal elment wrt $\subset$. (Note: we wanted to always consider proper filters. This is ok since we have (and can exclude) a minimal elment of $\mathcal{B}$). Consider such a maximal F. Suppose $x \notin F, \overline{x} \notin F$. Consider $F \cup \{x\}$ and close to obtain a filter (or if x is the least element of $\mathcal{B}$, adjoin $\overline{x}$ instead). This gives a new bigger filter, contradiction. Thus, if F is maximal then $\forall x \in \mathcal{B}$, $x \in F$ or $\overline{x} \in F$. Such filters are called ultrafilters.

In $\mathbb{N}$, a trivial example is $\{S \subset \mathbb{N} : 17 \in S\}$. But, writing down a sentence that picks out a nonprincipal ultrafilter is, to the lecturer's knowledge, impossible in any Boolean algebra; the lecturer or my intermediary is unsure whether this is actually a theorem.

Let $\{\mathcal{A}_i : i \in I\}$ be a family of structure of the same signature (i.e. with the same number and types of operations and relations). $\prod_{i\in I} \mathcal{A}_i$ is that structure whose carrier (aka underlying) set is the "cartesian" (or "direct") product $\phi_{i\in I} A_i$ of the carrier sets; the operations, relations etc. are defined pointwise.

Typically, we do this only when all the $\mathcal{A}_i \vDash T$ for some syntacticly simple T, e.g. T is algebraic if T is axiomatized by $(\forall \boldsymbol{x})(\bigwedge_{j\in J} t_j = s_j)$ where the t_j, s_j are words in the function letters of $\mathcal{L}(T)$; this is "universal closure of conjunction

of equations".

Birkhoff's theorem is that a theory T is algebraic iff the class of its models is closed under homomorphism, substructure and product; a proof will not be given in this course. There are many theorems of this form, relating syntactic and semantic definitions of some particular class of theories.

For $\prod_{i\in I}\mathcal{A}_i$ consider a filter $F\subset\mathcal{P}(I)$. This defines an equivalence relation on $\prod_{i\in I}A_i$ the carrier set of $\prod_{i\in I}\mathcal{A}_i$ by $f\sim_F g$ if $\{i:f(i)=g(i)\}\in F$. This opens up the possibility of quotient structures, but what are the relations on the quotient? $\frac{\prod_{i\in I}\mathcal{A}_i}{F}\vDash R([f],[g])$ iff $\{i:R(f(i),g(i))\}\in F$. These are called reduced products (wrt F); they are "nothing special" unless F is ultra. In that case the quotient "reduced product" is called an ultraproduct

Jerzy Łoś's Theorem: Let $\{A_i:i\in I\}$ be a family of structures all with the same signature, and let $\mathcal{U}\subset\mathcal{P}(I)$ be an ultrafilter. Then $\frac{\prod_{i\in I}\mathcal{A}}{\mathcal{U}}\vDash\phi$ iff $\{i:A_i\vDash\phi\}\in\mathcal{U}$; the proof is inductive, not given here.

An interesting special case: when all the factors are the same, we write $\frac{\mathcal{A}^I}{\mathcal{U}}$ and call this an ultrapower. We have an embedding of $\mathcal{A}$ into $\frac{\mathcal{A}^I}{\mathcal{U}}$ by $a\mapsto[\lambda i.a]_{\mathcal{U}}$ (the lecturer clearly expected people to immediately grasp, but I did not, that we are viewing A^I as the set of functions $I\to A$, which is correct (since that's what it is), but still somewhat nonintuitive). This is an elementary embedding: if $(\forall\boldsymbol{a})(\mathcal{A}\vDash\phi(\boldsymbol{a})\Rightarrow\mathcal{B}\vDash\phi(\boldsymbol{i}(\boldsymbol{a})))$. If i is actually the identity (so $\mathcal{A}\subset\mathcal{B}$) we say $\mathcal{A}$ is an elementary substructure of $\mathcal{B}$, $\mathcal{A}\prec\mathcal{B}$. E.g. $\langle\mathbb{Q},\leq\rangle$ is an elementary substructure of $\langle\mathbb{R},\leq\rangle$, and $\langle\mathbb{Q},+,\leq,0\rangle\prec\langle\mathbb{R},+,\leq,0\rangle$, but this is not so if we also include multiplication (we would then have "too many zeroes").

Suppose for ϕ $\mathcal{A}\vDash\phi$ iff $\mathcal{B}\vDash\phi$. We say $\mathcal{A}\equiv\mathcal{B}$, $\mathcal{A}$ and $\mathcal{B}$ are elementarily equivalent. (There are nontrivial questions to be asked about this notion, e.g. until recently it was unknown whether the free groups on 2 and 3 generators are elementarily equivalent).

If every finite subset of T has a model then T has a model. We have shown this via consistency: the former implies everyfinite subset of T is consistent, thus T is consistent, and then finally we get the result; we would like a more direct proof, which we can do by using ultraproducts:

Let Δ be a set of sentences in predicate calculus such that every finite $\Delta'\subset\Delta$ has a model. Then Δ has a model: Let S be the set of finite subsets of Δ (which we shall see later by the name of $\mathcal{P}_{\aleph_0}(\Delta)$). Let $X_s=\{t\in S:s\subset t\}$. For each s pick a model $\mathcal{M}_s\vDash s$. Consider the family $\{X_s:s\in S\}$. This has the finite intersection property: every finite subset of it has nonemty intersection, which the lecturer claims means it can be extended to an ultrafilter $\mathcal{U}\subset\mathcal{P}(S)$. Now consider the ultraproduct $\frac{\prod_{s\in S}\mathcal{M}_s}{\mathcal{U}}$. This is a model of Δ, because for every $\phi\in\Delta$, $X_{\{\phi\}}$ is one of the sets that generated the ultrafilter $\mathcal{U}$. Note that this proof did not require the assumption that the language was countable.

A model $\mathcal{M}$ is said to be saturated if whenever $\phi,\phi_2,\phi_3,\ldots$ all with the same number of arguments are "finitely satisfiable", i.e. $\forall$ finite $X\subset\mathbb{N}\exists\boldsymbol{x}:\bigwedge_{i\in X}\phi_i(\boldsymbol{x})$ there always $\exists\boldsymbol{x}:\bigwedge_{i\in\mathbb{N}}\phi_i(\boldsymbol{x})$.

Ehrenfeucht-Mostowski - although not the first to use the notion of a set of indiscernibles, they were the first to realise it was a useful, general one. For $\mathcal{M}$ a structure, $\langle I,<\rangle\subset M$ is a set of indiscernibles iff $\forall\phi(\ldots)\in\mathcal{L}(\mathcal{M}),\mathcal{M}\vDash\phi(x_1,\ldots,x_n)\Leftrightarrow\mathcal{M}\vDash\phi(y_1,\ldots,y_n)$ for $x_1<x_2<\cdots<x_n$ and $y_1<y_2<\cdots<y_n$ - "$\mathcal{M}$ cannot distinguish between ordered tuples".

Let T be a theory with infinite models and $\langle I,<\rangle$ any total order. Then T

has a model in which $\langle I, < \rangle$ is embedded as a SOI. The original proof, using Ramsey's theorem, is: Add to $\mathcal{L}(T)$ names c_i for every member of I. Add axioms to supply order information about the c_i. Call this theory $T^\star$. Now add (infinitely many) axioms to say the c_i are a SOI. Call this theory T^I. We will show that every finite fragment of T^I is consistent, then done: consider such a fragment T'. It mentions only finitely many constants, say c_1, c_3, c_4, c_5, and only finitely many predicates $\phi_1, \phi_2, \phi_3, \phi_4$. Let $\mathcal{M}$ be a model of $T^\star$ (which exists by compactness). These predicates, plus $<$, divide up $[\{c_i : i \in I\}^M]^m$ (where $[X]^m = \{Y \subset X : |Y| = m\}$) (where m is the supremum of the arities of $\phi_1, \phi_2, \phi_3, \phi_4$), according to where each is true or false (so in fact they will divide it into 2^n pieces where there are n predicates) (there is some irrelevant faff for when the ϕ_i have different arities). By Ramsey's theorem there is a monochromatic set of size ¿ the set of constants under consideration (in fact infinite).

A directed set is a set $(X, \leq)$: $\forall x, y \in X \exists z \in X : x, y \leq z$. A set is κ-directed if $\forall x_1, \ldots, x_\kappa \in X \exists z : \forall i, x_i \leq z$. A directed system is a family $\langle \mathcal{M}_i : i \in I \rangle$ of structures, where $\langle I \leq \rangle$ is a directed set, such that for $i < j \in I \exists f_{ij} : \mathcal{M}_i \hookrightarrow \mathcal{M}_j$ and the embeddings commute. This gives us the concept of a <u>limit</u> structure (in the terminology of the field, a direct limit; in category-theoretic language, a colimit); we could define this explicitly, but it is more enlightening to consider abstractly. ThIf the embeddings f_{ij} are all elementary, so are the embeddings $\mathcal{M}_i \hookrightarrow \mathcal{M}_\infty$. Without elementarity, all we get is preservation of [at this point I had enough of the lecturing of this course].