First-Order Logic

Harry Sha

Fall 2026

First-Order Logic

First-order logic is another type of logic that you're probably familiar with at this point.

We want to use it to talk about "objects".

Terms

Terms represent objects.

- Constants $c_1, ...$ are terms.
- Variables $x_1, ...$ are terms.
- If f is an n-ary function, and $t_1, ..., t_n$ are terms, then $f(t_1, ..., t_n)$ is a term.

Nothing else is a term.

Formulas

Formulas are statements about objects that might be true or false.

- If R is an n-ary relation and $t_1, ..., t_n$ are terms, then $R(t_1, ..., t_n)$ is a formula.
- If s and t are terms, then $s = t$ is a formula.
- If A and B are formulas, then $\neg A$, $(A \vee B)$, and $(A \wedge B)$ are formulas.
- If A is a formula and x is a variable, $\forall x A$, $\exists x A$ are formulas.

Nothing else is a formula.

Formulas of the first two types are called *atomic formulas*.

First-Order Languages

A first-order language fixes the following

- Constants $c_1, ...,$
- Functions $f_1, ...,$
- Relations $R_1, ...$

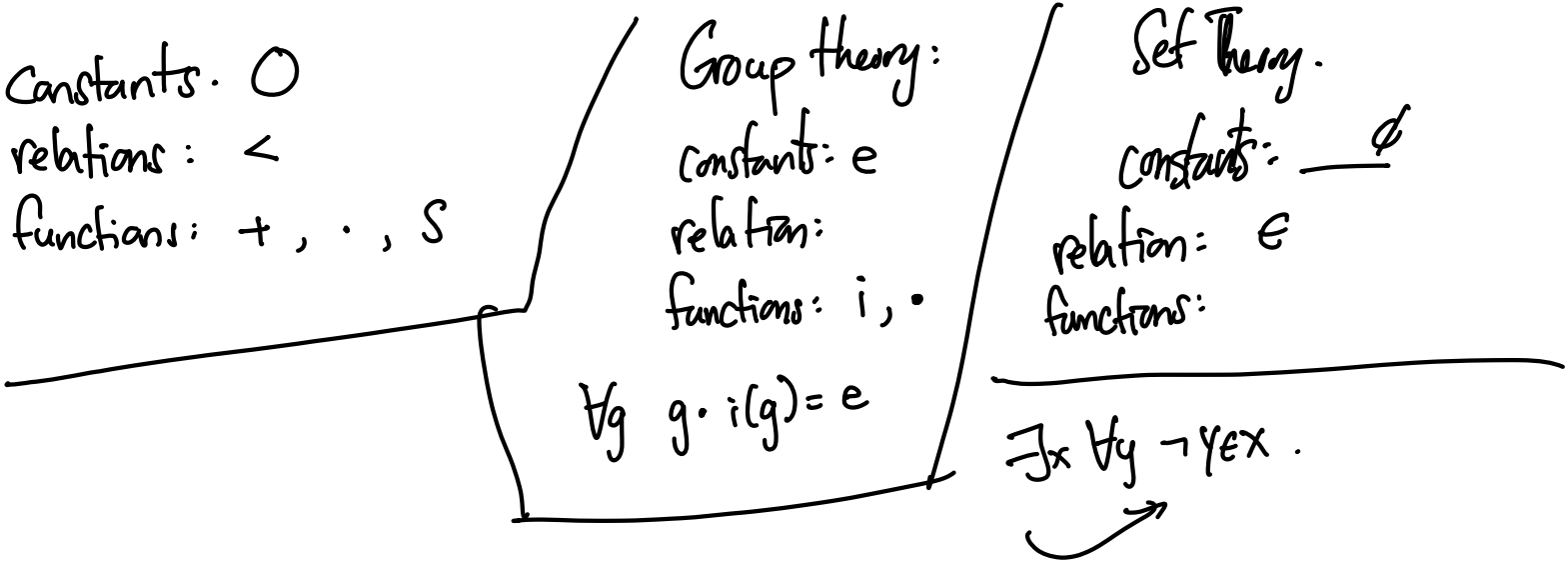
} vocabulary.

Each function and relation comes with its arity.

$=$ is a special case, and does not count as a relation.

We use $A \to B$ for $\neg A \vee B$, and $A \leftrightarrow B$ for $(A \to B) \wedge (B \to A)$.

Examples: Group Theory, Set Theory, Arithmetic



Ground terms

$x+0$

$0+S(0)$

For a term t, define $\mathrm{Var}(t)$ to be the set of variables in t.

A term with $\mathrm{Var}(t) = \emptyset$ is called ground. I.e. t only depends on constants and functions.

Section 1

Semantics

Structure

A structure is the first-order logic analogue of a truth assignment. It gives meaning to a formula.

In first-order logic, we're talking about objects, so the structure needs to interpret the constants, variables, and functions. It also needs to assign truth values to formulas.

Structure

For a given first-order language $\mathcal{L}$, an $\mathcal{L}$-structure must assign values to all the terms.

A structure $\mathcal{M}$ defines a non-empty set, M, called the domain.

It also gives values to all the constant symbols, and interprets all the functions:

- For each constant symbol c, $c^{\mathcal{M}} \in M$.
- For each n-ary function symbol f, $f^{\mathcal{M}} : M^n \to M$.

Variable assignments

If A has any free variables (to be defined), the truth of A also depends on how those free variables are assigned.

A variable assignment in $\mathcal{M}$ is a function $s : \mathrm{Var} \to M$.

$x = 0$

Value of a term

A structure, combined with a variable assignment, is enough to assign a value to each term t. We denote this value $t^{\mathcal{M}}[s]$.

Note t is a term (it's a string), and $t^{\mathcal{M}}[s] \in M$ is how it is interpreted by $\mathcal{M}, s$.

Truth for formulas

A structure also needs to define "truth" for formulas.

Again we can look at the recursive definition of formulas, and see that the structure really only needs to decide what to do with the relations R.

For each n-ary relation R, $R^{\mathcal{M}} \subseteq M^n$ is the interpretation of the relation R.

Note: the $=$ relation is always interpreted as actual equality in M.

$S : Var \to M$.

$M = \{0, 1, 2\}$.

$x = 0$

$S(x) = 0$

$M, s \models x = 0$.

Satisfies

Fix a structure $\mathcal{M}$, an assignment s, and a formula A. Write $\mathcal{M}, s \models A$ to mean $\mathcal{M}$ satisfies A under the variable assignment s.

Exercise (in class): Define this!

A can be: $R(t_1, \ldots, t_n)$, $t_1 = t_2$, $A \vee B$, $A \wedge B$, $\neg A$, $\forall x\, A$, $\exists x\, A$

How would you define $\mathcal{M}, s \models A$ in each case?

$\mathcal{M}, s \models R(t_1, \ldots, t_n)$ if $(t_1^{\mathcal{M}}[s], \ldots, t_n^{\mathcal{M}}[s]) \in R^{\mathcal{M}}$

$\mathcal{M}, s \models t_1 = t_2$ if $t_1^{\mathcal{M}}[s] = t_2^{\mathcal{M}}[s]$

↑ this is the character "=".

← = in the domain

$\mathcal{M}, s \models A \vee B$ if $\mathcal{M}, s \models A$ or $\mathcal{M}, s \models B$

$A \wedge B$

$\neg A$ are similar.

$\mathcal{M}, s \models \exists x A$ if there is some $b \in M$, s.t. $\mathcal{M}, s[x \to b] \models A$

$\mathcal{M}, s \models \forall x A$ if every $b \in M$, s.t. $\mathcal{M}, s[x \to b] \models A$

Sentences

A sentence is a formula with no free variables.

If A is a sentence, then either every assignment in $\mathcal{M}$ satisfies A or none does.

When A is a sentence we can write $\mathcal{M} \models A$ (omitting s).

Example

Take the vocabulary $\mathcal{L} = \{0, S, +, \cdot, <\}$.

Is the statement $\forall x \exists y (y < x)$ true? Bad question! Truth depends on the structure.

$\mathcal{M}$. $M = \mathbb{Z}$, take the "standard" interpretations of $S, +, \cdot, <$.

$\exists x \, S(S(x)) = 0$

Section 2

Properties of (sets of) formulas

The following definitions are essentially identical to the case of propositional logic!

Satisfiable

Let Σ be a set of formulas over the language $\mathcal{L}$. Σ is satisfiable if there are an $\mathcal{L}$-structure $\mathcal{M}$ and an assignment s such that $\mathcal{M}, s \models B$ for every $B \in \Sigma$.

When every formula in Σ is a sentence, then satisfaction doesn't depend on s. So we write $\mathcal{M} \models \Sigma$, and say $\mathcal{M}$ is a model of Σ.

Logically implies

For a set of formulas Σ, and formula A, we write $\Sigma \models A$ to mean that every structure $\mathcal{M}$, and assignment s that satisfies Σ, also satisfies A. If Σ is a singleton, we omit the set notation. E.g. $B \models A$ instead of $\{B\} \models A$.

If $B \models A$ and $A \models B$, we say that A and B are logically equivalent and we write $A \equiv B$.

A formula A is logically valid, written $\models A$ if $\emptyset \models A$. I.e. $\mathcal{M}, s \models A$ for every structure $\mathcal{M}$, and every assignment s.

Example: Not valid (but can be true!)

$$\forall x \exists y (x < y)$$

Example: Valid

$$(\forall x P(x)) \rightarrow P(y)$$

Useful fact

For every formula A:

- $\mathcal{M}, s \models \neg\forall x \neg A$ if and only if $\mathcal{M}, s \models \exists x A$, and
- $\mathcal{M}, s \models \neg\exists x \neg A$ if and only if $\mathcal{M}, s \models \forall x A$.

Convince yourself of this after reading the rules for satisfaction.

Section 3

LK

Sequents

Recall, sequents look like this

$$A_1, ..., A_k \Rightarrow B_1, ..., B_\ell$$

And recall that $(\mathcal{M}, s)$ satisfies a sequent if it falsifies one of the A_is or satisfies one of the B_is.

The rules of LK

Same as PK, except we need to handle the quantifiers.

How do we prove...

$\forall x.A$?

Pick arbitrary y, and then show $A[y/x]$.

$\exists x.A$?

Identify a witness, t and then show $A[t/x]$

How do we prove...

$\forall x.A$. Pick a fresh variable y. Prove the formula obtained by replacing every free occurrence of x in A with y.

$\exists x.A$. Find a term t that is free for x in A, and prove the formula obtained by replacing every free occurrence of x in A with t.

Substitution

For a formula A, define $A[t/x]$ to be the formula obtained by replacing every free occurrence of x in A with t, provided this substitution does not capture any variables.

Capture

y is not free for x in A.

Suppose we were trying to prove

$$A = \exists x \forall y E(x, y)$$

"There's some vertex that is adjacent to every vertex".

If we proposed that y is our witness and tried to directly substitute we'd get the following: $\forall y E(y, y)$, which is not what we intended! Notice that y got "captured" by the quantifier!

There is an easy fix, which is to simply rename the bound variable e.g. $y \mapsto z$.

Avoiding capture

To avoid capture, we only define the substitution $A[t/x]$ if t is free for x in A, as defined on the next slide.

A few definitions...

The scope of the quantifier in $\forall x A$ or $\exists x A$ is A.

An occurrence of a variable x inside a term of a formula is *bound* if it lies within the scope of a quantifier $\forall x$ or $\exists x$. Otherwise, it's *free*.

Let $\mathrm{FV}(A)$ be the set of variables that appear free somewhere in A.

A term t is *free for a variable x in a formula A* if no free occurrence of x in A lies within the scope of a quantifier binding a variable that occurs in t.

↳ this just means I can do $A[t/x]$, w/o worrying about capture.

The quantifier rules

$\forall$ intro — instantiation

$$\forall L\ \frac{A[t/x], \Gamma \Rightarrow \Delta}{\forall x\, A, \Gamma \Rightarrow \Delta} \qquad \forall R\ \frac{\Gamma \Rightarrow \Delta, A[y/x]}{\Gamma \Rightarrow \Delta, \forall x\, A}$$

generalization.

$\exists$ intro

$$\exists L\ \frac{A[y/x], \Gamma \Rightarrow \Delta}{\exists x\, A, \Gamma \Rightarrow \Delta} \qquad \exists R\ \frac{\Gamma \Rightarrow \Delta, A[t/x]}{\Gamma \Rightarrow \Delta, \exists x\, A}$$

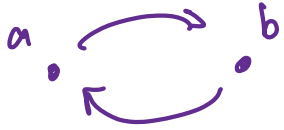
- t, y must be free for x in A
- y must not occur anywhere in the conclusion (below the rule)

Notes

- It can make more sense to read the rules bottom to top.
- The variable y is called the eigenvariable

The ‘free for’ condition

$$\frac{\exists y\ R(y,y) \Rightarrow \exists y\ R(y,y)}{\forall x\ \exists y\ R(x,y) \Rightarrow \exists y\ R(y,y)}\ \forall L \text{ (invalid)}$$



The 'freshness' condition

$$\frac{P(y) \Rightarrow P(y)}{P(y) \Rightarrow \forall x\, P(x)}$$ $\forall R$ (invalid).

Example: $\forall x(P(x) \wedge Q(x)) \Rightarrow \forall x P(x)$

$$\dfrac{\dfrac{\dfrac{\dfrac{P(y) \Rightarrow P(y)}{P(y), Q(y) \Rightarrow P(y)}\,W}{P(y) \wedge Q(y) \Rightarrow P(y)}\,\wedge L}{\forall x(P(x) \wedge Q(x)) \Rightarrow P(y)}\,\forall L}{\forall x(P(x) \wedge Q(x)) \Rightarrow \forall x(P(x))}\,\forall R$$

We write $\vdash_{LK} \varphi$ if there is a proof with endsequent φ.

As before, a proof of a formula A is a proof of the sequent $\Rightarrow A$.

Soundness and Completeness

We want to show $\models \varphi \iff \vdash \varphi$.

As with propositional logic, we'll show $\Sigma \models \varphi \iff \Sigma \vdash \varphi$ afterwards, for sets Σ of sentences.

Section 4

Soundness

Soundness

Recall that to prove soundness we show that all the axioms are valid, and then each rule preserves validity. Then by induction, the endsequent will also be valid!

We already know that all the PK rules are sound, so we just need to show that the quantifier rules are also sound.

First, a useful lemma...

Substitution Lemma. If t is free for x in A, then

$$\mathcal{M}, s \models A[t/x] \text{ iff } \mathcal{M}, s[x \mapsto t^{\mathcal{M}}[s]] \models A$$

semantic. $t^{\mathcal{M}}[s] \in M$.

↳ this is a string.

Note. t is a term. $t^{\mathcal{M}}[s]$ is an element of the domain!

In English: Assigning x an object a is the same as substituting t for every free occurrence of x in A, where t is a term that gets interpreted as a.

Proof

Structural induction on A.

Base case. A is atomic. $A = R(x)$

$\mathcal{M}, s \models R(x)[t/x] \iff$

$\mathcal{M}, s \models R(t). \iff$ by def of $\models$.

$\mathcal{M}, s[x \mapsto t^{\mathcal{M}}[s]] \models R(x).$

Proof

Inductive step. $A = B \vee C$

$\mathcal{M}, s \vDash B \vee C\ [t/x] \iff$

$\mathcal{M}, s \vDash B[t/x]$ or $\mathcal{M}, s \vDash C[t/x]$.

$\Downarrow$ (IH)

$\mathcal{M}, s[x \to t^{\mathcal{M}}[s]] \vDash B$. or $\mathcal{M}, s[x \to t^{\mathcal{M}}[s]] \vDash C$.

$\iff$ (def $\vDash$).

$\mathcal{M}, s[x \mapsto t^{\mathcal{M}}[s]] \vDash B \vee C$.

$A = \forall y\, B$

* (note $y \notin Vars(t)$ b/c "t is free for x in $\forall y\, B$" assumption).

$\mathcal{M}, s \models \forall y\, B[t/x]$

$\mathcal{M}, s[y \to a] \models B[t/x]$ for all $a \in M$.

$\mathcal{M}, s_a[x \mapsto t^{\mathcal{M}}[s_a]] \models B$ for all $a \in M$ (IH)

↳ $s_a = s[y \to a]$

$\mathcal{M}, s_a[x \mapsto t^{\mathcal{M}}[s]] \models B$ (*) for all $a \in M$.

$\mathcal{M}, s[x \mapsto t^{\mathcal{M}}[s]] \models \forall y\, B$.

Another lemma

If y is not free in a formula A, changing the value assigned to y does not change the truth of A.

Local Soundness $\exists$ R

$$\frac{\Gamma \to \Delta,\ A[t/x]}{\Gamma \to \Delta,\ \exists x\, A}\ (\exists R)$$

assume $\mathcal{M}, s \models A[t/x]$

$\Rightarrow \mathcal{M}, s[x \mapsto t^{\mathcal{M}}[s]] \models A$ (subs lemma)

$\Rightarrow \mathcal{M}, s \models \exists x\, A$. (def $\models$).

Local Soundness $\forall$ R

$$\frac{\Gamma \to \Delta, A[y/x]}{\Gamma \to \Delta, \forall x A}$$

assume $\mathcal{M}, s \models A[y/x]$

$\Rightarrow \mathcal{M}, s[x \mapsto s(y)] \models A$

$\mathcal{M}, s \models \forall x A$.

y doesn't appear anywhere below the line; it's not in Γ, nor in Δ, nor in A.

hence the assignment of y doesn't change any truth values.

Other cases

Exercise!

Section 5

Completeness

Completeness

Again, the proof echoes that of the completeness proof for propositional logic.

We start with the things we want to show and then progressively remove the connectives using the PK rules.

This doesn't work directly for quantifiers. If we had $\forall x A$, we might want to instantiate A multiple times!

Example

$\forall x(P(x) \to P(f(x))), P(c) \Rightarrow P(f(f(f(c))))$.

$$\frac{\forall x[P(x) \to P(f(x))], \forall x[P(x) \to P(f(x))], P(c) \Rightarrow P(f^3(c)),}{\forall x(P(x) \to P(f(x))), P(c) \Rightarrow P(f(f(f(c))))} C$$

Modified rules

For convenience, we'll introduce two modified rules (which can be derived from the original rules (exercise)).

$$\forall L^* \quad \frac{A[t/x], \forall x\, A, \Gamma \Rightarrow \Delta}{\forall x\, A, \Gamma \Rightarrow \Delta}$$

$$\exists R^* \quad \frac{\Gamma \Rightarrow \Delta, \exists x\, A, A[t/x]}{\Gamma \Rightarrow \Delta, \exists x\, A}$$

Backward proof search

We need to deal with the quantifiers.

We'll need some fresh variables $p_0, p_1, \ldots$ to use as eigenvariables.

Backward proof search

Starting at the thing we want to prove, we continually simplify our formulas. Since there are infinitely many terms to try for each $\forall$ on the left and each $\exists$ on the right, we need to be careful in order not to get stuck!

Example

Suppose our language has a unary function S, and a constant 0 and we're trying to prove: $\forall x P(x) \Rightarrow \exists x P(x)$

$$\dfrac{\dfrac{\forall x\, P(x),\ P(0), P(S(0)) \Rightarrow \exists x P(x)}{\forall x\, P(x),\ P(0) \Rightarrow \exists x\, P(x)}\ \forall L^*}{\forall x\, P(x) \Rightarrow \exists x\, P(x)}\ \forall L^*$$

Obligations

Whenever we see a non-atomic formula in a sequent, there's a corresponding rule to simplify the formula. Applying that rule to the formula is called an obligation. For $\forall L$ and $\exists R$, there are infinitely many obligations (one for each term).

Alg

Make p_0 and the root's free variables available as parameters.

- For round $n = 1, 2, ...$
  - For each open leaf φ present at the start of round n.
  - If the same atom occurs on both sides, close the branch.
  - Else, find the first n formulas in φ and first n available terms (T_n), process each of those finitely many obligations (for the rules where you have to instantiate terms, only use terms from T_n). If we need to introduce an eigenvariable, take the first unused p_i. If child branches are created, they inherit the obligations and parameters from their parent.
  - If every leaf is closed, output the proof!

Strategy

We claim that if φ is valid, then the algorithm finds a proof. To show this, we'll prove the contrapositive. If the algorithm doesn't find a proof, then φ is not valid. I.e. some $\mathcal{M}, s$ does not satisfy φ.

We'll construct such a model $\mathcal{M}, s$.

The countermodel

Suppose the search for a proof does not terminate, then the search tree is infinite. Since the search tree is finitely branching (each rule produces at most 2 branches), there must be an infinite branch.

Infinite Branch

Consider the infinite branch $S_0, S_1, \ldots$ where $S_n = \Gamma_n \Rightarrow \Delta_n$. Treat Γ_n and Δ_n as sets of formulas.

- Assume $\Gamma_n \subseteq \Gamma_{n+1}$, and $\Delta_n \subseteq \Delta_{n+1}$
- Assume $\Gamma_\infty = \bigcup_{n \in \mathbb{N}} \Gamma_n$
- Assume $\Delta_\infty = \bigcup_{n \in \mathbb{N}} \Delta_n$

Saturation

"The search tries everything eventually".
processes all obligations eventually.

Lemma (Saturation). On an infinite open branch the search:

- No atomic formula belongs to both Γ_∞ and Δ_∞.
- If $\neg A \in \Gamma_\infty$, then $A \in \Delta_\infty$. If $\neg A \in \Delta_\infty$, then $A \in \Gamma_\infty$.
- If $A \wedge B \in \Gamma_\infty$, then both $A, B \in \Gamma_\infty$. If $A \wedge B \in \Delta_\infty$, then at least one of A, B belongs to Δ_∞.
- If $A \vee B \in \Gamma_\infty$, then at least one of A, B belongs to Γ_∞. If $A \vee B \in \Delta_\infty$, then both $A, B \in \Delta_\infty$.

Saturation

Lemma (continued).

- If $\forall x A \in \Gamma_\infty$, then $A[t/x] \in \Gamma_\infty$ for every term t over the branch parameters.
- If $\forall x A \in \Delta_\infty$, then $A[y/x] \in \Delta_\infty$ for some branch variable y.
- If $\exists x A \in \Gamma_\infty$, then $A[y/x] \in \Gamma_\infty$ for some branch variable y.
- If $\exists x A \in \Delta_\infty$, then $A[t/x] \in \Delta_\infty$ for every term t over the branch parameters.

Proof of the lemma

SKIP!

Building the countermodel on this infinite branch

We want to construct a model $\mathcal{T}$, and assignment s_0 that satisfies every formula in Γ_0, and falsifies every formula in Δ_0.

This part is confusing, but very clever.

The domain

$T = \{ c_1, P_0, f(c_1), ff(c), P_1, g(P_1) \ldots \}$

The domain for our countermodel is going to be the set of terms available on this branch. Let's call this set T. They include any branch parameters, constants, and functions on those.

The objects in this domain are the strings corresponding to the terms.

The term structure

The term structure $\mathcal{T}$ has domain T. We then need to say how $\mathcal{T}$ interprets constants, functions, and relations. Essentially, $\mathcal{T}$ just sends everything to itself:

- $c^{\mathcal{T}} = c$
- $f^{\mathcal{T}}(t_1, ..., t_k) = f(t_1, ..., t_k)$
- $R^{\mathcal{T}} = \{(t_1, ..., t_k) : R(t_1, ..., t_k) \in \Gamma_\infty\}$

Define the variable assignment s_0 as follows. $s_0(y) = y$ for every branch parameter y, and map variables that are not branch parameters to p_0.

The term structure is a countermodel

We now show that $\mathcal{T}, s_0$ is a countermodel. I.e.

- For each $A \in \Gamma_\infty$ $\mathcal{T}, s_0 \models A$
- For each $A \in \Delta_\infty$, $\mathcal{T}, s_0 \not\models A$

$R(t_1, \ldots, t_k)$ ✓

$A = B \vee C$. $A \in \Gamma_\Phi$.

WTS $\mathcal{T}, s_0 \models B$ or $\mathcal{T}, s_0 \models C$.

True by saturation + IH.

Equality

There's a slight caveat with the previous proof. We didn't address the equality relation!

In defining the countermodel, we were able to pick the interpretation of each relation R. However, the equality relation must always be interpreted as actual equality.

We'll come back to this.

Premises

$B_1, \ldots B_k, \Gamma \Rightarrow$

$\Rightarrow B$.

We first deal with premises.

Similar to PK, if Σ is a set of sentences, and φ is a sequent, write $\Sigma \vdash_{LK} \varphi$ if there's an LK proof of φ where leaves of the form $\Rightarrow B$ where $B \in \Sigma$ are allowed.

Prior to this, we just showed that $\vdash \varphi \iff \models \varphi$. We now want $\Sigma \vdash \varphi \iff \Sigma \models \varphi$, where Σ is a set of sentences.

Proof

modify the proof search alg. s.t. that in round n, I add the n^{th} element of Σ to Γ.

Suppose there was an infinite branch. then $\Sigma \subseteq \Gamma_\infty$.

Then we have $\Upsilon, s_0 \vDash \Sigma$, but

$$\Upsilon, s_0 \nvDash \Gamma_0 \Rightarrow \Delta_0 .$$

so $$\Sigma \nvDash \Gamma_0 \Rightarrow \Delta_0$$

Compactness

For first-order sentences, the following three statements are equivalent, and all hold:

1. If Σ is unsatisfiable, then some finite subset of Σ is unsatisfiable.
2. If $\Sigma \models A$, then $\Sigma_0 \models A$ for some finite $\Sigma_0 \subseteq \Sigma$.
3. If every finite subset of Σ is satisfiable, then Σ is satisfiable.

$\Sigma \models A \Rightarrow \Sigma \vdash A \Rightarrow \Sigma_0 \vdash A \Rightarrow \Sigma_0 \models A$

(completeness) (finiteness of proofs) (soundness).

Proof

Note

This is a slightly different order to how we presented compactness in propositional logic.

There we started by proving compactness and then used that to justify completeness from assumptions.

Here we started with a more powerful search algorithm, and used it to conclude both compactness and completeness at the same time!

We could have done the same in propositional logic.

Equality

To handle equality, we'll introduce axioms that make the interpretation of $=$ an equivalence relation compatible with all functions and relations. Taking the quotient by this relation then makes equality actual identity in the quotient domain.

For now, think of $=$ as any old relation.

Universal Closure

Let $x_1, \ldots, x_n$ be the free variables of a formula B.

The *universal closure* of B is the sentence

$$\forall x_1 \cdots \forall x_n \, B.$$

If B is already a sentence, its universal closure is B itself.

For example, the universal closure of $R(x, y) \to R(y, x)$ is

$$\forall x \forall y \, (R(x, y) \to R(y, x)).$$

Equality Axioms

Let $\mathrm{Eq}_{\mathcal{L}}$ contain the universal closures of:

$$x = x, \qquad x = y \to y = x, \qquad (x = y \wedge y = z) \to x = z;$$

for every k-ary function symbol f,

$$\bigwedge_{i=1}^{k} x_i = y_i \to f(x_1, \ldots, x_k) = f(y_1, \ldots, y_k);$$

and for every k-ary relation symbol $R \neq =$,

$$\bigwedge_{i=1}^{k} x_i = y_i \to \big(R(x_1, \ldots, x_k) \leftrightarrow R(y_1, \ldots, y_k)\big).$$

Lemma. Let $\mathcal{M} \models \mathrm{Eq}_{\mathcal{L}}$, and write $\sim$ for the relation interpreting $=$ in $\mathcal{M}$.

Then $\sim$ is an equivalence relation, and the quotient $\mathcal{M}/\!\sim$ is a well-defined $\mathcal{L}$-structure. For every formula A and assignment s,

$$\mathcal{M}, s \models A \iff \mathcal{M}/\!\sim, [s] \models A.$$

Here $[s](x) = [s(x)]$, and equality in the quotient is actual identity of equivalence classes.

$[x] = \{y \in M : x \sim y\}$. $\quad M/\!\sim \;= \{[x] : x \in M\}$.

Proof

$$M/\sim \;=\; \{[x] : x \in M\}.$$

$$c^{M/\sim} = [c^M]$$

$$f^{M/\sim}([t_1], \ldots, [t_k]) = [f(t_1^M[s], \ldots, t_k^M[s])]$$

$$R^{M/\sim}([t_1], \ldots, [t_k]) \Leftrightarrow R^M(t_1, \ldots, t_k).$$

LK proof with equality

For a set Σ of sentences, we define

$$\Sigma \vdash_{\mathrm{LK}=} A \quad \Longleftrightarrow \quad \Sigma \cup \mathrm{Eq}_{\mathcal{L}} \vdash_{\mathrm{LK}} A,$$

where $=$ is an ordinary binary relation symbol on the right.

When $\Sigma = \emptyset$, write simply $\vdash_{\mathrm{LK}=} A$.

Soundness and Completeness with Equality

Theorem. For a set Σ of sentences and a sentence A,

$$\Sigma \vdash_{\mathrm{LK}=} A \quad \Longleftrightarrow \quad \Sigma \models A,$$

where $=$ has its genuine identity interpretation on the semantic side.

Proof

Section 6

Theories

Definitions

A *theory* T in a vocabulary $\mathcal{L}$ is a set of $\mathcal{L}$-sentences.

Its *model class* is

$$\mathrm{Mod}(T) = \{\mathcal{M} : \mathcal{M} \models B \text{ for every } B \in T\}.$$

Its *semantic closure* is

$$\mathrm{Cn}(T) = \{A : A \text{ is a sentence and } T \models A\}.$$

The theory of a structure and theorems of a theory

The *theory of a structure* $\mathcal{M}$ is

$$\mathrm{Th}(\mathcal{M}) = \{A : A \text{ is a sentence and } \mathcal{M} \models A\}.$$

The set of *theorems* of T is

$$\mathrm{Thm}(T) = \{A : A \text{ is a sentence and } T \vdash A\}.$$

Use LK with equality when the vocabulary includes $=$.

Soundness and completeness give $\mathrm{Thm}(T) = \mathrm{Cn}(T)$.

Consistent, satisfiable, complete

A theory T is:

- *Satisfiable* if there is $\mathcal{M}$ such that $\mathcal{M} \models T$.
- *(Syntactically) consistent* if there is no sentence B such that both $T \vdash B$ and $T \vdash \neg B$.
- Complete if, for every sentence A, either $T \vdash A$ or $T \vdash \neg A$.

Model existence

semantic

Corollary. A first-order theory T is satisfiable if and only if it is syntactically consistent. } syntax.

Let C be some contradiction (like $A \wedge \neg A$).

Suppose T is consistent. $T \nvdash C$

$\Rightarrow T \not\models C$ (completeness)

$\exists \mathcal{M}$ s.t. $\mathcal{M} \models T$, $\mathcal{M} \not\models C$

$\Rightarrow$ satisfiable.

Suppose T is not consistent $T \vdash C$

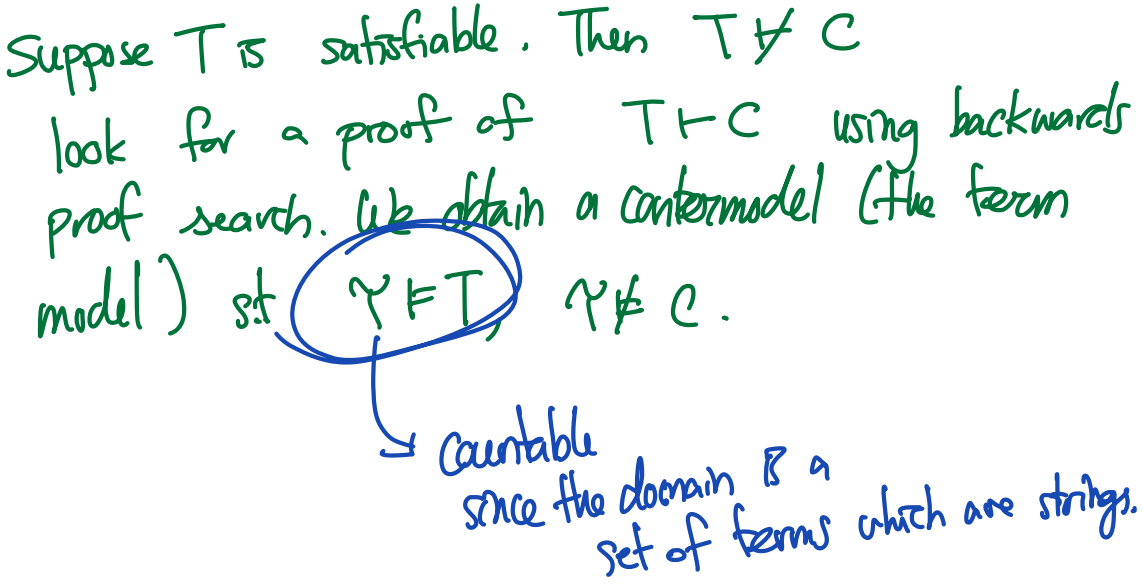
$\Rightarrow T \models C$

$\Rightarrow$ every model $\mathcal{M} \models T$, also has $\mathcal{M} \models C$.

$\Rightarrow$ no model has $\mathcal{M} \models T$ since C is a contradiction

Downward Löwenheim–Skolem Theorem

Theorem. Every satisfiable theory in a countable first-order vocabulary has a model whose domain is countable.



Countable

Recall, countable means countably infinite or finite.

It's possible to come up with a theory for which all models have for example, two elements (Exercise).

Arbitrarily Large Finite Models

Theorem. If a theory T has finite models of arbitrarily large size, then T has an infinite model.

That is, if for every $n \geq 1$ there is a finite model of T with at least n elements, then there is an infinite model of T.

$X = T \cup \{ c_i \neq c_j : i \neq j \}$. , c_i s are all new .

Claim : every finite subset of X is satisfiable.

$\Rightarrow$ every finite subset has finitely many c_is .

take a large enough model for T s.t. I can assign a different element to each c_i .

$\Rightarrow$ compactness X is sat. w/ a countable, infinite model .