Completeness for FOL

Overview

- ✓ Adding Witnessing Constants
- ✓ The Henkin Theory
- ✓ The Elimination Theorem
- The Henkin Construction

Lemma 12

- This lemma assures us that our construction of $\mathfrak{M}_h$ works for the atomic sentences.
- That is, $\mathfrak{M}_h$ will make an atomic sentence true if and only if *h* assigns TRUE to that atomic sentence.
- **Lemma 12**. If c ≡ c', d ≡ d', and *h*(R(c, d)) = TRUE, then *h*(R(c', d')) = TRUE.

Lemma 12

- **Proof**. By Exercise 19.8, the following is a tautological consequence of the Henkin theory *H*:

  $$(R(c, d) \wedge c = c' \wedge d = d') \rightarrow R(c', d')$$

- Since *h* assigns everything in *H* TRUE
- And it assigns TRUE to each conjunct of

  $$R(c, d) \wedge c = c' \wedge d = d',$$

- It must also assign TRUE to $R(c', d')$.

Lemma 13

- **Lemma 13.** For any sentence **S** of L_H, $\mathfrak{M}_h \models$ S if and only if h(**S**) = TRUE.
- **Remarks.** We have explicitly defined the structure $\mathfrak{M}_h$ to ensure the claim is true for atomic sentences.
- Further, truth assignments work the same way on truth-functional connectives as the definition of truth in a first-order structure.
- The only possible problems are the quantifiers.
- As we will see, they are taken care of by the quantifier axioms in *H*.
- Notice that the quantifier ∀ is not handled directly, but indirectly through the deMorgan sentences in *H*:

  $\neg\forall x P(x) \leftrightarrow \exists x \neg P(x)$

Lemma 13

- The proof will be by induction.
- Due to the above, we cannot use the length or number of logical operations of a wff.
- As it would count count $\forall x P(x)$ as simpler than the sentence $\exists x \neg P(x)$,
- We define a different measure of complexity for wffs.
- We define the complexity of an atomic wff to be 0
- The complexity of $\neg P$ and $\exists x P$ to be one greater than the complexity of P.
- The complexity of $P \wedge Q$, $P \vee Q$, and $P \rightarrow Q$ to be one greater than the maximum of that of P and Q
- The complexity of $\forall x P$ to be three greater than that of P.

Examples of wffs and their Complexities

wff	complexity
$\mathsf{Small}(\mathsf{x})$	0
$(\mathsf{x} = \mathsf{a})$	0
$\neg(\mathsf{x} = \mathsf{a})$	1
$\mathsf{Small}(\mathsf{x}) \rightarrow \neg(\mathsf{x} = \mathsf{a})$	2
$\neg(\mathsf{Small}(\mathsf{x}) \rightarrow \neg(\mathsf{x} = \mathsf{a}))$	3
$\exists \mathsf{x} \neg(\mathsf{Small}(\mathsf{x}) \rightarrow \neg(\mathsf{x} = \mathsf{a}))$	4
$\forall \mathsf{x} (\mathsf{Small}(\mathsf{x}) \rightarrow \neg(\mathsf{x} = \mathsf{a}))$	5

Proof of Lemma 13

- **Proof**. By induction on the complexity of sentences.
- Base case. When the complexity is 0 the lemma is true by the way we defined the structure $\mathfrak{M}_h$.
- Assume that the lemma holds for all sentences of complexity ≤ k and let **S** have complexity ≤ k + 1.
- Inductive case. There are several cases to consider, depending on the main connective or quantifier of **S**.
- We prove one of the truth-functional cases, as these are all similar, and then both of the quantifier cases.

Proof of Lemma 13

- **Case 1.** Suppose S is P ∨ Q. If $\mathfrak{M}_h \models S$, then at least one of P or Q is true.
- Assume that P is true.
- Since the complexity of S is ≤ k+1, the complexity of P is ≤ k.
- By induction hypothesis, *h*(P) = TRUE.
- But then *h*(P ∨ Q) = TRUE, as desired.
- The proof in the other direction is similar.

Proof of Lemma 13

- **Case 2**. Suppose that **S** is ∃xP(x).
- We need to show that $\mathfrak{M}_h$ |= ∃xP(x) if and only if *h* assigns the sentence TRUE.
- Assume first that the sentence is true in $\mathfrak{M}_h$.
- Since every object in the domain is denoted by some constant, there is a constant **c** such that $\mathfrak{M}_h$ |= P(**c**).
- The complexity of this sentence is less than that of **S.**
- By our induction hypothesis h(P(c)) = TRUE.
- Recall that our theory *H* contains the sentence P(c) → ∃xP(x)

Proof of Lemma 13

- As such, *h* assigns this sentence TRUE.
- By the truth table for →, *h* assigns true to ∃xP(x), as desired.
- The reverse direction of this case is very similar.
- It uses the Henkin witnessing axiom for P(x).
- Assume that *h* assigns true to ∃xP(x).
- We need to show that $\mathfrak{M}_h$ |= ∃xP(x).
- Recall that *h* assigns TRUE to the witnessing axiom ∃xP(x) → P(cP(x))
- By the truth table for →, *h* assigns TRUE to $P(c_{P(x)})$.
- By induction, this sentence is true in $\mathfrak{M}_h$.
- But then ∃xP(x) is true as well.

Proof of Lemma 13

- **Case 3. L**et us assume that **S** is ∀xP(x).
- We need to prove that this sentence is true in $\mathfrak{M}_h$ if and only if *h* assigns the sentence TRUE.
- Assume first that **S** is true in $\mathfrak{M}_h$.
- In this case, ∃x ¬P(x) is false in $\mathfrak{M}_h$.
- By induction, *h* assigns FALSE to this sentence.
- Recall that *H* contains the sentence ¬∀xP(x) ↔ ∃x ¬P(x)
- From this it follows that *h* assigns false to ¬∀xP(x) and hence true to ∀xP(x), as desired.
- The proof in the other direction is entirely similar.

Function symbols

- If there are function symbols in the original language, we have to explain how to interpret them in our structure.
- Suppose, for example, that our language contains a one-place function symbol **f**.
- How should we define its interpretation *f*?
- In particular, if **d** is some constant symbol, what equivalence class should *f*([**d**]) be?
- What comes to our rescue here is the witnessing constant for the sentence ∃x [f(d) = x]

Function symbols

- We can define $f([\mathbf{d}])$ to be the equivalence class $[c_{f(d)=x}]$ of the witnessing constant $c_{f(d)=x}$.
- $\exists x\,[f(d) = x] \rightarrow f(d) = c_{f(d)=x}$ is in *H*
- As such, it is not hard to check that all the details of the proof work out pretty much without change.
- This completes our filling in of the outline of the proof of the Completeness Theorem.

Review: Putting Everything Together

- Assume that *T* and *S* are all from the original language *L* and that *S* is a first-order consequence of *T*.
- We want to prove that $T \vdash S$.
- By assumption, there can be no first-order structure in which all of $T \cup \{\neg S\}$ is true.
- By the Henkin Construction lemma, there can be no truth assignment *h* which assigns TRUE to all sentences in $T \cup H \cup \{\neg S\}$.

Review: Putting Everything Together

- If there were, then the first-order structure $\mathfrak{M}_h$ would make $T \cup \{\neg S\}$ true.
- Hence *S* is a tautological consequence of $T \cup H$.
- The Completeness Theorem for propositional logic tells us there is a formal proof *p* of *S* from $T \cup H$.
- The Elimination Theorem tells us that using the quantifier rules, we can transform *p* into a formal proof *p* of *S* from premises in *T* .
- Hence, T |- S, as desired.