

THINGS TO KNOW FOR THE IN CLASS EXAM

The in-class part of the exam will ask you to provide a combination of definitions and proofs of theorems.

To remind you of something which is probably obvious: The list of definitions and theorems are short enough that you can just memorize them by rote. You could do worse things (like not learn them at all), but over the long term you'll learn them better if you understand them, rather than memorize them.

1. OVERVIEW

1.1. Definitions: You need to know the definitions of what it means for a set to be definable, what a theory is, what it means for a theory to be complete, and it means for a collection of formulas to contain witnesses.

I'm not expecting you to know the proof of the completeness theorem, but you should know how to construct a model of a consistent theory, even if you don't know how to prove that it really is a model. In other words, you should know how to construct what the book calls the *term structure*, \mathfrak{T}^Φ , corresponding to a given collection of formulas Φ .

You should know the definition of elementary equivalence, and the definition of an isomorphism (and automorphism) of models.

From the computability theory part of the course, you should know the definition of decidable, and the definition of enumerable. You don't need to know the definition of register machine.

I'm not going to ask any of the definitions from the first exam, but you should review them before looking at the definitions below, since the definitions below build off of the definitions from the first part of the class.

1.2. Theorems. You should know how to prove the following: the compactness theorem, the Upward-Löwenheim-Skolem Theorem, and the undecidability of the halting problem. In addition, you should be able to prove that decidable sets are enumerable, and that if both a set and its complement is enumerable then the set is in fact decidable. You should be able to explain why the set of programs that halt is enumerable even though it is not decidable.

Note that of the above, the undecidability of the halting problem is by far the hardest.

2. THE DEFINITIONS

definable. Let \mathfrak{M} be a model with universe M . A subset $S \subseteq M^n$ is *definable* iff there is some $\varphi_{x_1, \dots, x_n}$ such that an n -tuple (m_1, \dots, m_n) is an element of S if and only if there is some assignment of variables β such that $(\mathfrak{M}, \beta \frac{m_1, \dots, m_n}{x_1, \dots, x_n}) \models \varphi$.

A comment: We think of this as “ S is *defined by* φ means that (m_1, \dots, m_n) is in S iff φ is true of (m_1, \dots, m_n) ”. If it is not clear why the above definition captures this idea, you should look back at the definition of $(\mathfrak{M}, \beta) \models \varphi$.

theory. Fix a language, L . A *theory* is a set of sentences in L .

complete. A theory is *complete* if for a φ , T proves either φ or $\neg\varphi$.

contains witnesses. A theory *contains witnesses* iff for any formula of the form $\exists x\varphi(x)$, there is a term, t , such that T proves $\varphi(t)$.

the term structure. Let L be a language, and let Φ be a theory in L . Let T be the set of terms of L . Define the following equivalence relation on T . Say $t_1 \sim t_2$ iff Φ proves that $t_1 = t_2$. Then the *term structure* associated to Φ (written \mathfrak{T}^Φ) has universe T modulo the equivalence relation \sim . As a matter of notation, we will write $[t]$ to mean “the equivalence class in \mathfrak{T}^Φ to which t belongs”. Make \mathfrak{T}^Φ a model in the language L as follows: For each constant c in L interpret c as $[c]$. For each function f , for each $[t_1], \dots, [t_n]$ in \mathfrak{T}^Φ define $f([t_1], \dots, [t_n])$, as $[f(t_1, \dots, t_n)]$. Finally, say that $R([t_1], \dots, [t_n])$ holds iff Φ proves $R(t_1, \dots, t_n)$.

Th(\mathfrak{M}). The collection of sentences φ such that $\mathfrak{M} \models \varphi$ is written as $Th(\mathfrak{M})$.

elementarily equivalent. We say that \mathfrak{M} is *elementarily equivalent* to \mathfrak{N} iff $Th(\mathfrak{M}) = Th(\mathfrak{N})$.

isomorphism. Let \mathfrak{M} and \mathfrak{N} be models in the language L . We say that a map $f : M \rightarrow N$ is an *isomorphism* iff

- (1): f is a bijection
- (2): f preserves constants: that is, for each c , $f(c) = c$.
- (3): f preserves functions in L : that is, for each function $g(x_1, \dots, x_n)$ in L , $f(g(x_1, \dots, x_n)) = g(f(x_1), \dots, f(x_n))$.
- (4): f preserves relations in L : that is, for each relation $R(x_1, \dots, x_n)$ in L , $R(x_1, \dots, x_n)$ iff $R(f(x_1), \dots, f(x_n))$.

decidable. A set, W , is *decidable* if there is a register machine P such that $P : w \rightarrow$ “yes” for $w \in W$, and $P : w \rightarrow$ “no” otherwise.

2.1. enumerable. A set W is *enumerable* iff there is a register machine P such that for any $w \in W$, P (on empty input) eventually prints out w , and any w that P prints is in W .

THEOREMS

Theorem 2.1. (Compactness) *Suppose that every finite subset of a theory T has a model. Then T has a model.*

Proof. Since each finite subset of T has a model, each finite subset of T is consistent. Since each proof is of finite length, if you could prove a contradiction from T , you could do it with a finite subset of T . Since you can't, T is consistent. Thus the completeness theorem implies that T has a model. □

Theorem 2.2. (Upward-Löwenheim-Skolem) *If T is consistent and has some model of size at least n for each $n \in \mathbb{N}$, and κ is some cardinal, then T has models of cardinality greater than or equal to κ .*

Proof. This is not the strongest possible version of the theorem. The book has a stronger version. However, what is stated above captures the basic idea: any theory T has arbitrarily large models. And the proof is not difficult.

Recall that a cardinal is just the size of a set. So we must prove

The proof is just an application of the compactness theorem. Suppose you want to show that T has a model of size greater than or equal to some particular cardinal, say $\kappa = |I|$. Now one expands L by adding $|I|$ -many new constants, say $L' := L \cup \{c_i : i \in I\}$. Now expand T by saying that each of these new constants is distinct. That is, $T' := T \cup \{c_i \neq c_j : i \neq j\}$. All that remains is to show that every finite subset of T' has a model, since then by the compactness theorem, T' will have a model, and it is clear that any model of T' will have at least $|I|$ -many elements.

Take some finite subset of T' , say T_F . T_F will only mention finitely many of the new constants. Say it mentions n of them. Pick some model \mathfrak{M} of T of size at least n , and interpret each of the finitely many new constants as a different element of \mathfrak{M} . Thus $\mathfrak{M} \models T_F$, as desired. \square

Theorem 2.3. (Undecidability of the Halting Problem) *The set $\Pi_{halt} := \{w_P | P : \square \rightarrow halt\}$ is undecidable.*

Proof. We first show that the set $\Pi'_{halt} := \{w_P | P : w_P \rightarrow halt\}$ is undecidable. Suppose that it is decidable by P_0 . Then let P_1 halt iff its input is a program not in Π'_{halt} . Now consider the question of whether P_1 is in Π'_{halt} or not.

Suppose it is. Then P_1 halts when given itself as input, thus P_1 is not in Π'_{halt} .

On the other hand, suppose P_1 is not in Π'_{halt} . Then P_1 does not halt when given itself as input. Thus it is in Π'_{halt} .

Now that we have shown that Π'_{halt} is undecidable, we have to reduce the question of decidability of Π_{halt} to that of Π'_{halt} . Take a program P and create P' as follows. The first thing that P' does is load w_P into R_0 . Then P' runs P . Thus $P' : \square \rightarrow halt$ iff $P : w_P \rightarrow halt$. That is, $P' \in \Pi_{halt}$ iff $P \in \Pi'_{halt}$.

Now suppose that one has a program P_{halt} that decides Π_{halt} . We create a program, P_2 , that decides Π'_{halt} as follows: P_2 takes a program P , changes it to P' as above, and then runs P_{halt} on P' . Finally, P_2 then gives whatever answer P_{halt} gives for P' .

Thus P_2 says “yes” iff $P' : \square \rightarrow halt$ iff $P : w_P \rightarrow halt$. But we have shown that this is impossible. \square

Theorem 2.4. *Decidable implies enumerable*

Proof. Take $W \subset A^*$ decidable and P deciding it. Now let P_1 be the program that generates each element of A^* in lexicographic order, runs P on each one, outputting the element iff P says “yes”. Clearly P enumerates W . \square

Theorem 2.5. *W is decidable iff both W and W^C is enumerable.*

Proof. Suppose that W is decidable. The W^C is also decidable: If P decides W , then let P' be the program that outputs “no” when P says “yes” and vice versa. Then P' decides W^C . Thus by the preceding theorem, both W and W^C are enumerable.

Now suppose that W and W^C are both enumerable, say by P_1 and P_2 respectively. Let P be the program that takes an input w , runs both P_1 and P_2 , and outputs “yes” if P_1 outputs w , and outputs “no” if P_2 outputs w . Since w must be in the output of either P_1 or P_2 , P will eventually halt on any w . \square

Theorem 2.6. *The set Π_{halt} described above is enumerable.*

Proof. Let P_{halt} be the following program: at stage n , P_{halt} generates in lexicographical order the first n programs and runs k th program through the first $n - k + 1$

steps. That is, it runs program 1, n steps, program 2 $n - 1$ steps, all the way to program n , which it runs for 1 step. Then it outputs any program that halts. Then it proceeds to stage $n + 1$

To show that P_{halt} eventually outputs each program that halts, consider an arbitrary program P . Say P halts after it runs for m steps, and suppose that P is i th program in lexicographic order. Thus at stage $m + i - 1$, P_{halt} will run P for $(m + i - 1) - i + 1 = i$ steps, discover that it halts, and output it. \square