A3 Categorial Grammar

1. Introduction

In this chapter, we present the salient features of categorial grammar, which is employed throughout this book to do syntactic analysis.

2. Basic Ideas – Noun Phrases, Sentences, Functors

Categorial grammar classifies grammatical expressions according to category. The grammatical categories consist of primitive and derivative categories. There are two primitive categories.

N	noun phrase
S	sentence

In addition to these two primitive categories, there are also derivative categories, which classify the huge variety of *functors* (officially defined below).

In Simple Categorial Grammar, the *categories* are inductively defined as follows.

Df
N is a category;
S is a category;
if K_0, K_1, ..., K_m are categories, where $m \geqslant 0$,
then the following is also a category: $\langle K_1, ..., K_m, K_0 \rangle$;
nothing else is a category.

We will depict the category $\langle K_1, ..., K_m, K_0 \rangle$ in various ways, including the following.

$(K_1, ..., K_m \rightarrow K_0)$
$(K_1 + ... + K_m \rightarrow K_0)$
$(K_1 + ... + K_m) \rightarrow K_0$

Also, parentheses are often dropped, the rules of omission paralleling those for sentential connectives.

Associated with derivative categories are the class of grammatical expressions called *functors*, which are defined as follows.

Df
A functor is a grammatical expression with blanks (gaps, places, slots), which yields a grammatical expression of a particular category, when its blanks are filled with grammatical expressions of the appropriate categories.

The basic idea is this.

Suppose functor ϕ has category (K_1+ ...+ $K_m \rightarrow K_0$). Then
ϕ has m blanks, which require fill-in expressions of category K_1, ..., K_m, respectively, and when so filled, ϕ yields an expression of category K_0.

3. Monadic, Polyadic, Anadic Functors

Df
Where k is any natural number (0, 1, 2, ...), a *k-place functor* is a functor with k blanks (places).

Alternative names:

monadic	:	1-place
dyadic	:	2-place
triadic	:	3-place
polyadic	:	2-place or 3-place or …

Some functors are not specifically 1-place, 2-place, or any particular place. These are called *anadic* functors. The prefix ‘an’ means ‘without’, so an anadic functor is a functor without “adic”. It is arguable that conjunction and disjunction (especially exclusive disjunction) are examples of anadic connectives. The following is our official definition.

Df
An *anadic functor* is a syntactic expression with an open-ended blank that, when filled with any number (≥ 0) of expressions, all of a single particular category, results in an expression of a particular category.

A simple example of an anadic functor is anadic conjunction, which takes any number of sentences and yields a sentence. This is categorially depicted as follows.

$$S^* \rightarrow S$$

Here, the symbol ‘*’ indicates that the functor takes any number of sentences and produces a sentence.

4. Simple Functors

Df

A *homogeneous functor* is a functor whose input must all be of the same category.

Homogeneous functors allow for a simplification of notation as follows.

$S1 \rightarrow S$	$=_{df}$	$S \rightarrow S$
$S2 \rightarrow S$	$=_{df}$	$(S+S) \rightarrow S$
$S3 \rightarrow S$	$=_{df}$	$(S+S+S) \rightarrow S$
etc.		

For example, a functor of category [S2➔S] takes two sentences and produces a sentence.

Df

A *rank-one* functor is a functor whose input *and* output must be one of the primitive categories – N and S.

This is a special case of the more general notion of rank, which is officially defined in Section 12.

Df

A *simple* functor is a homogeneous rank-one functor.

Simple functors come in exactly four general varieties.

function signs (operators)	N*➔N; Nk➔N	takes zero or more noun phrases, and yields a noun phrase
connectives (sentential operators)	S*➔S; Sk➔S	takes zero or more sentences, and yields a sentence
predicates	N*➔S; Nk➔S	takes zero or more noun phrases, and yields a sentence
subnectives	S*➔N; Sk➔N	takes zero or more sentences, and yields a noun phrase

See Section 7 for examples of non-simple functors in elementary logic.

5. Zero-Place Functors

Curiously, the theory of grammatical categories allows for zero-place functors. For example, a zero-place connective takes zero-many(!) sentences, and produces a sentence. Similarly, a zero-place predicate takes zero-many noun phrases and produces a sentence, and a zero-place function sign takes zero-many noun phrases and produces a noun phrase. These are categorially depicted as follows.

S0➔S	N0➔S	N0➔N
∅➔S	∅➔S	∅➔N

Here, ∅ is the "null" category. [See Section 12 for official notation.]

Zero-place connectives do not occur naturally. The most prominent theoretical example of a zero-place connective is the contradiction symbol '✕' used primarily in derivations. Note, however, that '✕' has other theoretical uses; for example, one can prove that all truth-functional connectives are definable in terms of '→' and '✕'. [See Chapter 4.]

Zero-place function signs and zero-place predicates are also primarily of theoretical use. For example, one can treat proper nouns as zero-place function signs, and one can treat sentential constants (inherited from elementary SL) as zero-place predicates. Also, subject-less sentences (for example, 'it is raining') can be fruitfully analyzed using zero-place predicates.

6. Variable Binding Functors

An additional critical feature of *logical* syntax is *variable binding*. The following is the general definition.

Df

A *variable-binding functor* is a functor, at least one blank of which must be filled by a *variable*. Whenever a functor requires a variable as input, it *binds* that variable.

Variable-binding functors are well-known in first-order logic, and include the two quantifiers and the description operator. They appear in second-order logic as well, and include the second-order quantifiers and the lambda operator. Also, set theory provides a prominent example of a variable-binding functor – the set-abstract operator – {v: __}; filling the blank with a formula generates a singular term.

In order to accommodate variable-binding in categorial grammar, we introduce subcategories, described by the following definition.

Df

For every category K, there is a subcategory K(v) of variables associated with that category; any expression of category K(v) is automatically also an expression of category K; any functor that takes category K(v) as input binds the associated variable.

Generally, a given formal language utilizes only a few variable categories. For example, standard first-order logic uses just one variable category – the category of *individual variable*, which is a subcategory of N. Also, this sub-category is sufficiently important that we give it a special name – V_0.

$$V_0 \quad =_{df} \quad N(v)$$

7. Examples of Variable-Binding Functors

Elementary logic provides three well-known examples of variable-binding functors – the quantifiers and the description operator, which may be categorially described as follows.

Symbol	**Category**	**Grammatical Function**
$\forall$	$(V_0+S)\rightarrow S$	takes an individual variable, and a sentence, and yields a sentence
$\exists$	$(V_0+S)\rightarrow S$	takes an individual variable, and a sentence, and yields a sentence
ι	$(V_0+S)\rightarrow N$	takes an individual variable, and a sentence, and yields a noun phrase

Note that although these categories are rank-1, they are not homogeneous, so they are not simple according to our definition. For example, the functor $\forall$ takes an individual variable and a sentence as input and generates a sentence as output.

An alternative categorial account of these functors is given as follows.

Symbol	**Category**	**Grammatical Function**
$\forall$	$V_0\rightarrow(S\rightarrow S)$	takes an individual variable, and yields a one-place connective
$\exists$	$V_0\rightarrow(S\rightarrow S)$	takes an individual variable, and yields a one-place connective
ι	$V_0\rightarrow(S\rightarrow N)$	takes an individual variable, and yields a one-place subnective

According to the latter account, these functors are not rank-1 functors. In particular, $\forall$ and $\exists$ generate a one-place connective as output; and ι generates a one-place subnective as output. The notion of rank, and the rank-discrepancy between the two accounts of quantifiers, is reconsidered in Section 12.

The immediate lesson here is that categorial analysis is not cut-and-dried; a given linguistic entity may receive different categorial analyses. For a more thorough discussion, see Section 12.

8. More Examples of Complex Functors

Category	Grammatical Function
$N \rightarrow (N \rightarrow S)$	takes an expression of category N, and yields an expression of category $N \rightarrow S$; in other words, it takes a noun phrase and yields a 1-place predicate.
$(N \rightarrow S) \rightarrow (N \rightarrow S)$	takes an expression of category $N \rightarrow S$, and yields an expression of category $N \rightarrow S$; in other words, it takes a predicate and yields a predicate; these are sometimes called predicate adverbs.
$N \rightarrow [(N \rightarrow S) \rightarrow (N \rightarrow S)]$	takes something of category N, and yields an expression of category $(N \rightarrow S) \rightarrow (N \rightarrow S)$; in other words, it takes a noun phrase and yields a predicate adverb.
$[(N \rightarrow S) \rightarrow (N \rightarrow S)] \rightarrow [(N \rightarrow S) \rightarrow (N \rightarrow S)]$	takes an expression of category $(N \rightarrow S) \rightarrow (N \rightarrow S)$, and yields an expression of category $(N \rightarrow S) \rightarrow (N \rightarrow S)$; in other words, it takes a predicate adverb and yields a predicate adverb.
$V_0 + S \rightarrow (N \rightarrow S)$	takes an individual variable (binding it), and a sentence, and yields an expression of category $(N \rightarrow S)$, which is to say a one-place predicate.
$V_0 + V_0 + S \rightarrow (N + N \rightarrow S)$	takes two individual variables (binding them), and a sentence, and yields an expression of category $(N + N \rightarrow S)$, which is to say a two-place predicate.

9. Alternative Categorial Descriptions – 1

In an earlier section, we described quantifiers as having the following category.

$$V_0 + S \rightarrow S$$

In other words, according to this account, a quantifier functor '$\forall$' takes two input, one individual variable, and one formula, and generates a formula.

An alternative rendering of '$\forall$' describes it as a rank-2 functor, whose category is:

$$V_0 \rightarrow (S \rightarrow S).$$

According to this account, '$\forall$' is a functor that takes a single input – an individual variable – and generates an expression of category ($S \rightarrow S$), which is to say a one-place connective (or sentential adverb).

The difference between these two accounts is largely theoretical; the question is whether there is an autonomous intermediate grammatical expression '$\forall x$'; it seems plausible that there is.

The trick we just used on '$\forall$' can be applied to all polyadic functors. For example, the usual logical analysis of verbs like 'respects' treats them as two-place predicates – whose official category is

$$(N + N) \rightarrow S.$$

An alternative categorial analysis of such expressions is

$$N \rightarrow (N \rightarrow S).$$

According to this analysis, 'respects' takes a single noun phrase, and generates a one-place predicate, which in turn takes a single noun phrase and generates a sentence. For example, the sentence

Jay respects Kay

is analyzed at first as follows.

subject:	Jay	N (atomic)
predicate:	respects Kay	$N \rightarrow S$

The resulting non-atomic expression 'respects Kay' is then analyzed as follows.

functor:	respects(.)	$N \rightarrow (N \rightarrow S)$
argument:	Kay	atomic N

Algebraically formulated, the sentence has the following form.

[respects(Kay)](Jay)

In other words,

respects(.)

takes

Kay

as input, and produces

[respects(Kay)](.).

This latter expression, in turn, is a functor that takes

Jay

as input, and produces

[respects(Kay)](Jay).

This is in contrast with the usual logical analysis,

respects(Jay, Kay),

according to which ‘respects(..)’ is a two-place functor that takes ‘Jay’ and ‘Kay’ (on equal footing), and generates a sentence.

10. Alternative Categorial Descriptions – 2

Sometimes a difference in categorial analysis is not particularly profound, as illustrated in the previous section. Other times, a difference may be more theoretically cogent. Consider the following sentence.

Jay believes that Kay is intelligent.

There are several plausible categorial analyses of this sentence; the following are a few that come to mind.

#1: believes[Jay, that(Kay is intelligent)]

[(N+N)→S]áN, (S→N)áSññ

functor (verb):	believes	N+N→S
subject:	Jay	N
object:	that Kay is intelligent	N*

<u>sub-analysis</u> (that Kay is intelligent):

subnective:	that ___	S→N
argument:	Kay is intelligent	S*

#2: believes that[Jay, Kay is intelligent]

{(N+S)→S}áN,Sñ

functor:	believes that	N+S→S
argument 1:	Jay	N
argument 2:	Kay is intelligent	S*

#3: [believes thatáJayñ](Kay is intelligent)

{[N→(S→S)]áNñ}áSñ

functor:	Jay believes that ___	(S→S)*
argument:	Kay is intelligent	S*

<u>sub-analysis</u> (Jay believes that):

functor:	____ believes that	N→(S→S)
argument:	Jay	N

#4: [believes that Kay is intelligent](Jay)

(N➔S)áNñ

functor (predicate): believes that Kay is intelligent $(N \to S)^*$
argument: Jay

sub-analysis (believes that Kay is intelligent):
left open at the moment!

In each case, a starred category just means that the expression is complex. In some cases, the complex expression has been further analyzed; in other cases, it has not.

11. Categorial Equivalence

Examples from the two previous sections strongly suggest that categorial analysis is not so cut-and-dried as we might hope. In general, a given sentence can be categorially analyzed in many different ways. One important theoretical question is:

> Under what circumstances are two categorial analyses equivalent (in some sense), and conversely, under what circumstances are two analyses genuinely different?

For example, I consider some of the examples in the previous two sections to be superficially different, not genuinely different, and I consider other examples to be genuinely different.

Towards clarifying this question, I propose several sufficient (not necessary) conditions for categorial equivalence, which are stated algebraically. In the following, the variables range over categories.

(e1) $(A+B) \to C \equiv (B+A) \to C$;
(e2) $[A+(B+C)] \to D \equiv [(A+B)+C] \to D$
(e3) $(A+B) \to C \equiv A \to (B \to C)$
(e4) $(\varnothing + A) \to B \equiv A \to B$
(e5) $\varnothing \to A \equiv A$
(e6) if $A \equiv B$, then $K[A/B]$
Here K is a category, and K[A/B] results from K by substituting A for B in any of its occurrences.

Note that we have some logical consequences of these conditions, supposing from the outset that $\equiv$ is an equivalence relation.

(1) $A \to (B \to C) \equiv B \to (A \to C)$

Proof:

(1)	$A \rightarrow (B \rightarrow C) \equiv (A+B) \rightarrow C$	e3+Sym≡
(2)	$(A+B) \rightarrow C \equiv (B+A) \rightarrow C$	e1
(3)	$(B+A) \rightarrow C \equiv B \rightarrow (A \rightarrow C)$	e3
(4)	$A \rightarrow (B \rightarrow C) \equiv B \rightarrow (A \rightarrow C)$	1,2,3,Eq≡

Examples:

$$(V_0+S) \rightarrow S \equiv V_0 \rightarrow (S \rightarrow S)$$

$$(N+N) \rightarrow S \equiv N \rightarrow (N \rightarrow S)$$

$$\begin{aligned}(N+N+N) \rightarrow S &\equiv N \rightarrow [(N+N) \rightarrow S] \\ &\equiv (N+N) \rightarrow (N \rightarrow S) \\ &\equiv N \rightarrow [N \rightarrow (N \rightarrow S)]\end{aligned}$$

We conclude this section by noting a formal similarity between categorical equivalence and logical equivalence. In particular, if we translate '+' as '&', and '➔' as '→', then the equivalences proposed above translate as logical equivalences. For example, the categorial equivalences

$$(A+B) \rightarrow C \equiv A \rightarrow (B \rightarrow C)$$
$$(A+B) \rightarrow C \equiv (B+A) \rightarrow C$$

translate respectively as the following logical equivalences:

$$(A\&B) \to C \equiv A \to (B \to C)$$
$$(A\&B) \to C \equiv (B\&A) \to C.$$

We should not get carried away, however. Reverse translations don't always work. For example, the logical equivalence

$$(A\&A) \to B \equiv A \to B$$

translates as

$$(A+A) \rightarrow B \equiv A \rightarrow B$$

which is *not* a categorial equivalence.

12. The Concepts of Rank and Order

We have already encountered the notion of *rank*, in the form of rank-one functors. In the present section, we generalize on this concept to include rank-2, rank-3, etc., and we also offer a related notion of *order*. The generalized notion of rank can be inductively defined as follows.

$$\boxed{\begin{array}{l} \text{rank}(\varnothing) = 0 \\ \text{rank}(N) = 0 \\ \text{rank}(S) = 0 \\ \text{rank}(K_1+K_2+\ldots+K_m \rightarrow K_0) = 1+\max\{\text{rank}(K_1), \ldots, \text{rank}(K_m), \text{rank}(K_0)\} \end{array}}$$

Examples:

rank(N➔S) =
1+max{rank(S), rank(N)} =
1+max{0, 0} =
1

rank(N➔(N➔S)) =
1+max{rank(N), rank(N➔S)} =
1+max{0,1} =
2

rank((N➔S)➔S) =
1+max{rank(N➔S)}, rank(S)} =
1+max{2, 1} =
2

The theoretical uses of the notion of rank are limited. The main problem is that different rank functors can nevertheless be categorially equivalent. For example,

(N+N)➔S

is categorially equivalent to

N➔(N➔S).

but

rank[(N+N)➔S] = 1

whereas

rank[N➔(N➔S)] = 2.

In order to deal with this problem, it is useful to formulate a notion of level – which we call *order* – that is sensitive to categorial equivalence. In particular, the following is a key desideratum.

Des
If functors ϕ_1 and ϕ_2 are equivalent, then order(ϕ_1) = order(ϕ_2).

The following turns out to be our official definition of *order*.

order(∅) = 0
order(N) = 0
order(S) = 0
order(K_1+K_2+…+K_m➔K_0) = max{1+max{order(K_1) , …, order(K_m)}, order(K_0)}

Examples:

$\text{order}(N \rightarrow S)$ =
$\max\{1+\max\{\text{order}(S)\}, \text{order}(N)\}$ =
$\max\{1+\max\{0\}, 0\}$ =
$\max\{1, 0\}$ =
1

$\text{order}(N \rightarrow (N \rightarrow S))$ =
$\max\{1+\max\{\text{order}(N)\}, \text{order}(N \rightarrow S)\}$ =
$\max\{1+\max\{0\}, 1\}$ =
$\max\{1, 1\}$ =
1

$\text{order}((N \rightarrow S) \rightarrow S)$ =
$\max\{1+\max\{\text{order}(N \rightarrow S)\}, \text{order}(S)\}$ =
$\max\{1+\max\{1\}, 0\}$ =
$\max\{2, 0\}$ =
2

Theorems About Rank and Order

The following are two theorems about rank and order.

T1:
If output(K) = N or S, then rank(K) = order(K).

Here, *output* is defined in the obvious manner.

$\text{output}(K_1,\ldots,K_m,K_0) = K_0$

In other words, output(K) is the output category of category K

T2:
If $K_1 \equiv K_2$, then $\text{order}(K_1) = \text{order}(K_2)$

13. A Formal Account of Basic Categorial Grammar

In the present section, we present general categorial syntax. Our definitions subsume the definitions given for simple categorial grammar.

1. Basic Categories

Df

N is a category;
S is a category;

if K_0, K_1, ..., K_m are categories, where $m \geqslant 0$,
then the following is also a category: $\langle K_0, K_1, ..., K_m \rangle$;

nothing else is a category [unless it is anadic – see next section].

2. Anadic Categories

Df

If K and K_0 are categories, then so is: $(K^* \rightarrow K_0)$.

[Note carefully that K* is not itself a category; the *-sign is not categorimatic!]

3. Variable Categories

Df

For every category K, there is the sub-category K(v), which is the category of variables associated with category K.

4. Shorthand Definitions

$((K_1, ..., K_m) \rightarrow K_0)$	$=_{df}$	$\langle K_0, K_1, ..., K_m \rangle$
$\varnothing \rightarrow K$	$=_{df}$	$\langle K \rangle$
$(K_1+K_2) \rightarrow K_0$	$=_{df}$	$(K_1, K_2) \rightarrow K_0$
$(K_1+K_2+K_3) \rightarrow K_0$	$=_{df}$	$(K_1, K_2, K_3) \rightarrow K_0$
etc.		
K1	$=_{df}$	K
K2	$=_{df}$	(K+K)
K3	$=_{df}$	(K+K+K)
etc.		

[parentheses are optional when unnecessary for parsing.]

5. Functor Application

Df

if Φ is an expression of category $(K_1, ..., K_m \rightarrow K_0)$,
and $\eta_1, ..., \eta_m$ are expressions of category $K_1, ..., K_m$, respectively,
then $\Phi(\eta_1, ..., \eta_m)$ is an expression of category K_0.

if Φ is an expression of category $K^* \rightarrow K_0$,
and $\eta_1, ..., \eta_m$ are all expressions of category K,
then $\Phi(\eta_1, ..., \eta_m)$ is an expression of category K_0.

Here, $\Phi(\eta_1, ..., \eta_m)$ is the result of "applying" functor Φ to expressions $\eta_1, ..., \eta_m$ the details of which (prefix, infix, postfix, etc.) will depend upon the specific language.

6. A Note Concerning Functor Application

As indicated in the rules of application, the way functor application is *syntactically implemented* will depend upon the specific language. For example, in ordinary first-order logic, we have the following.

$\forall(\nu, \mathbb{F})$	=:	$\forall\nu\mathbb{F}$
$\exists(\nu, \mathbb{F})$	=:	$\exists\nu\mathbb{F}$
$=(\sigma, \tau)$	=:	$[\sigma = \tau]$
$\sim(\mathbb{F})$	=:	$\sim\mathbb{F}$
$\rightarrow(\mathbb{F}, \mathbb{G})$	=:	$(\mathbb{F} \rightarrow \mathbb{G})$

14. An Alternative Formal Account of Categories

In previous sections, we have presented the theory of categories in a way in which the symbols '+' and '$\rightarrow$' are merely shorthand forms of the official expressions. In the present section, we present an alternative theory of categories in which these notions are taken as fundamental (primitive). This theory also offers a formal account of order and categorial equivalence. Note, this theory does not include either anadic categories or variable categories, which are left as an exercise for the reader.

Vocabulary:

variables: K, K_0, K_1, etc.; Σ, Σ_1, Σ_2, etc.
proper nouns: N, S, $\varnothing$;
2-place function signs: +, $\rightarrow$, written in infix notation;
1-place predicates: …is a category, …is a category sum, written in postfix notation;
2-place predicate: $\equiv$, written in infix notation.

Axioms:

(c1) N is a category;
(c2) S is a category;
(c3) if Σ is a category sum, and K is a category, then $\Sigma \rightarrow K$ is a category;
(c4) nothing else is a category.

(s1) $\varnothing$ is a category sum;
(s2) every category is a category sum;
(s3) if Σ_1 and Σ_2 are category sums, then so is $\Sigma_1+\Sigma_2$;
(s4) nothing else is a category sum.

(a1) $\varnothing+\Sigma = \Sigma$;
(a2) $(\Sigma_1+\Sigma_2)+\Sigma_3 = \Sigma_1+(\Sigma_2+\Sigma_3)$;
(a3) $\Sigma_1+\Sigma_2 = \Sigma_2+\Sigma_1$;
(a4) $\varnothing \rightarrow K \equiv K$;
(a5) $K \equiv K$;
(a6) if $K_1 \equiv K_2$, then $K_2 \equiv K_1$;
(a7) if $K_1 \equiv K_2$, and $K_2 \equiv K_3$, then $K_1 \equiv K_3$;
(a8) $(\Sigma_1+\Sigma_2) \rightarrow K \equiv \Sigma_1 \rightarrow (\Sigma_2 \rightarrow K)$.

Parenthesis Convention:
Officially, every application of a two-place infix functor requires outer parentheses. On the other hand, we adopt the standard informal convention that if the resulting expression "stands alone", then the outer parentheses may be dropped. This convention is repeatedly applied in the axioms above.

Sortal Convention:
In axioms (o3)-(a8), the variables are understood as "sortal". In particular, the Σ-variables range over category sums, and the K-variables range over categories. These two classes overlap, of course, since every category is automatically a category sum.

Extremal Clause Convention:
Three of the axiom groups end with extremal clauses. This is a convenient shorthand for an inductive clause, the precise nature of which varies from situation to situation. For a clarification of extremal clauses, see Chapter 3 on mathematical induction.

The following are a few examples of theorems of this theory.

T1. If K is a category, then $\varnothing \rightarrow K$ is a category.

Proof. Suppose K is a category. By s1, $\varnothing$ is a category sum, so by c3, $\varnothing \rightarrow K$ is a category.

T2. If K_1 and K_2 are categories, then $K_1 \rightarrow K_2$ is a category.

Proof. Suppose K_1 and K_2 are categories. Then by s2, K_1 is a category sum, so by c3, $K_1 \rightarrow K_2$ is a category.

T3. $N \rightarrow (N+N)$ is neither a category sum nor a category.

Proof. This requires math induction; see Chapter 3.

T4. Let K be a category. Then there are category sums $\Sigma_1, \ldots, \Sigma_m$ such that: $K \equiv \Sigma_1 \rightarrow (\Sigma_2 \rightarrow (\ldots(\Sigma_m \rightarrow K_0)\ldots)$, where $K_0=N$ or $K_0=S$.

Proof. This requires induction; see Chapter 3.

T5. Let K_1 and K_2 be categories. Suppose K_1 is first-order, and suppose $K_1 \equiv K_2$. Then K_2 is first-order.

Proof. This theorem, which involves the notion of order (Section 12) requires induction; see Chapter 3.