Lecture V

Truth and Satisfaction in RE.

Remember that the satisfaction relation is a relation in two variables, S(A,s), which holds between a formula A and an assignment s sufficient for A just in case s satisfies A (in the case of RE, s assigns non-negative integers to the variables, since the intended interpretation of RE is the arithmetical interpretation). Since truth can be defined in terms of satisfaction, if RE could define its own satisfaction relation, RE would have its own truth predicate.

Some assignments related to formulae by the satisfaction relation are sequences of infinite length: the sequence $\{<x_1,0>, <x_2,1>, ...\}$ is an assignment of the value i-1 to the variable x_i; this assignment is naturally sufficient for any formula, and satisfies, e.g., all formulae of the form $x_i=x_i$. However, as Cantor showed, we could not code all infinite sequences of numbers by means of numbers, so the satisfaction relation for formulae of RE cannot be represented as a relation between numbers. However, for our purposes it is really unnecessary to contemplate the full satisfaction relation. It will be enough to be able to define within RE the satisfaction relation restricted to finite assignments, or even a relation Sat(a,s), which holds between a (code of a) formula A and a (code of a) finite function s which assigns non-negative integers to all the (codes of) variables appearing free in A and satisfies A in the obvious sense (thus, if Sat(a,s), s need not be a sequence, for its domain need not be an initial segment of the non-negative integers -nor an initial segment of the codes of variables-, and s need not be an assignment, for it can assign values to things other than codes of variables). Sat(a,s) will be the relation that we will show how to define in RE. In fact, we shall show, equivalently, that the set of Gödel numbers of pairs in Sat is r.e., using the Generated Sets Theorem. One way in which we can begin to see that this will be enough for our purposes is to note that if Sat can be defined in RE, then the truth predicate for RE can be defined in RE, since a sentence of RE is true just in case it is satisfied by some finite function.

We shall now undertake the proof of the following

Theorem: The satisfaction relation Sat(a,s) for formulae of RE is definable in RE, or, in other words, RE has its own satisfaction predicate.

We shall devote to this proof this lecture and the next one.

As we just said, in showing that the satisfaction relation for RE is r.e., we shall use the Generated Sets Theorem. What we shall show is that the *set* of (numbers coding) pairs G={[a, s]: s codes a function which is sufficient for and satisfies the formula whose Gödel number is a} is generated from an r.e. set by means of r.e. relations.

In order to prove our theorem it would be perhaps most natural to generate separately the set of formulae, then define in some way the notion of a function being sufficient for a formula, and finally generate the set of pairs <A,s> where s is sufficient for and satisfies the formula A, going through the clauses in the inductive definition of satisfaction for RE. However, we will generate this set in one fell swoop, so to speak, without having first to define the set of formulae and the relation of being a function sufficient for a formula.

We will now begin our specification of the basis set, and later we will define the generating relations. In order to show that the set that we will take as basis set is r.e., we will show first that the set of terms and the relation of denotation are r.e.

It is important to stress at this stage a delicate point in our coding scheme. Remember that in our coding scheme, in order to code formulae we code terms first, by coding the sequence of numbers that code the individual symbols appearing in a term (in the same order). Thus, a term $f^1_1f^1_1x_i$ will be coded by any code of the sequence {[1,[4,[1,1]]], [2,[4,[1,1]]], [3,[1,i]]}, and, *as a term*, x_i will be coded by any code of the sequence {[1,[1,i]]}. Then we code formulae using the same procedure, but now taking each term as if it was an individual symbol, a code for it being a code of the appropriate sequence. Thus, a formula $P^2_1x_ix_i$ will be coded by any code of the sequence {[1,[5,[2,1]]], [2,[1,[1,i]]], [3,[1,[1,i]]]}.

We now exhibit a formula Funct(s) which is true of a number if it codes a finite function:

$$\text{Funct}(s) =_{\text{df.}} (x \in s)(\exists m_1 \leq x)(\exists m_2 \leq x)(x=[m_1,m_2]) \land$$
$$(n_1 \leq x)(n_2 \leq x)(m \leq s)(([m,n_1] \in s \land [m,n_2] \in s) \supset n_1 = n_2).$$

With the help of Funct(s), we can give an alternative formula that shows that the relation holding between a sequence and its length is r.e.:

$$\text{Seql}(s,n) =_{\text{df.}} \text{Funct}(s) \land (i \leq n)(\mathbf{0}' \leq i \supset (\exists j)([i,j] \in s)).$$

We now specify a formula Num(m,n) which is true of a pair of numbers p,q if p is a code of a numeral that denotes number q:

$$\text{Num}(m,n) =_{\text{df.}} \text{Seql}(m,n') \land [n',[\mathbf{0}^{(2)},\mathbf{0}^{(1)}]] \in m \land (i \leq n)(\mathbf{0}' \leq i \supset [i,[\mathbf{0}^{(4)},[\mathbf{0}^{(1)},\mathbf{0}^{(1)}]]] \in m).$$

The first conjunct “says” that m is a sequence of length n+1; the second that the last pair of the sequence has as second element the code of the constant **0**, which, remember, is [2,1]; and the third conjunct “says” that all the other second elements of the sequence are codes of the symbol for successor, which is [4,[1,1]]. We can now give a formula Numeral(m) that defines the set of codes of numerals:

$$\text{Numeral}(m)=_{\text{df.}} (\exists n\leq m)(\text{Num}(m,n)).$$

The formula Vblt(m,v) will be true of two numbers p,q if p is a code of a term of the form $f^1_1...f^1_1x_i$, for a certain i, and v is the code of x_i (in this case we say that p is (a code of) a variable term ending in variable q):

$$\text{Vblt}(m,v)=_{\text{df.}} (\exists n\leq m)(\text{Seql}(m,n') \wedge (\exists i\leq m)(v=[\mathbf{0}^{(1)},i]) \wedge [n',v]\in m \wedge (j\leq n)(\mathbf{0}'\leq j \supset [j,[\mathbf{0}^{(4)},[\mathbf{0}^{(1)},\mathbf{0}^{(1)}]]]\in m)).$$

The first conjunct "says" that m is a sequence of length n+1; the second that v is the code of a variable; the third that the last value of the sequence is v; the fourth that all the values preceding v are codes of the symbol for successor. A formula similar to Numeral(m) then defines the set of codes of terms of the form $f^1_1...f^1_1x_i$:

$$\text{Vblterm}(m)=_{\text{df.}} (\exists v\leq m)(\text{Vblt}(m,v)).$$

It will be useful to introduce a formula Vbl(v) which is true of a number if it is the code of a variable:

$$\text{Vbl}(v)=_{\text{df.}} (\exists i\leq v)(v=[\mathbf{0}^{(1)},i]).$$

Finally, we can give a formula that defines the set of codes of terms:

$$\text{Term}(m)=_{\text{df.}} \text{Numeral}(m) \vee \text{Vblterm}(m)$$

(remember that in RE the only terms are numerals (in official notation) and variables preceded by a number of occurrences of the function letter f^1_1. If we had taken + and · as primitive function letters, there would have been more complicated terms. As it is, since ' is our only function symbol, things are much simpler).

We are now ready to define denotation. The formula Den(m,n,s) is true of a triple of numbers p,q,r if p is a term that denotes q with respect to assignment r:

$$\text{Den}(m,n,s)=_{\text{df.}} \text{Funct}(s) \wedge (\text{Num}(m,n) \vee (\exists v\leq m)(\text{Vblt}(m,v)\wedge(\exists p\leq m)(\text{Seql}(m,p'))\wedge(\exists q\leq s)([v,q]\in s\wedge q+p=n)).$$

The second disjunct of the second conjunct "says" that there is a variable v such that m is a variable term ending in v, m has length p+1 for a certain p and s assigns to v a number q such that, if you add to it 1 p times, you get n.

The formula $\text{Atf}^=(s)$ is true of a number if it codes an atomic formula of the form $P^2_1t_1t_2$, where t_1 and t_2 are terms of RE:

$$\mathrm{Atf}^{=}(s)=_{\mathrm{df.}}\ (\exists m_1\leq s)(\exists m_2\leq s)(\mathrm{Seql}(s,\mathbf{0}^{(3)})\wedge \mathrm{Term}(m_1)\wedge \mathrm{Term}(m_2)\wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(2)},\mathbf{0}^{(1)}]]]\in s\wedge [\mathbf{0}^{(2)},m_1]\in s\wedge [\mathbf{0}^{(3)},m_2]\in s).$$

The formula $\mathrm{Atf}^{A}(s)$ is true of a number if it codes an atomic formula of the form $P^3_1t_1t_2t_3$, where t_1, t_2 and t_3 are terms of RE:

$$\mathrm{Atf}^{A}(s)=_{\mathrm{df.}}\ (\exists m_1\leq s)(\exists m_2\leq s)(\exists m_3\leq s)(\mathrm{Seql}(s,\mathbf{0}^{(4)})\wedge \mathrm{Term}(m_1)\wedge \mathrm{Term}(m_2)\wedge \mathrm{Term}(m_3)\wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(3)},\mathbf{0}^{(1)}]]]\in s\wedge [\mathbf{0}^{(2)},m_1]\in s\wedge [\mathbf{0}^{(3)},m_2]\in s\wedge [\mathbf{0}^{(4)},m_3]\in s).$$

The formula $\mathrm{Atf}^{M}(s)$ is true of a number if it codes an atomic formula of the form $P^3_2t_1t_2t_3$, where t_1, t_2 and t_3 are terms of RE:

$$\mathrm{Atf}^{M}(s)=_{\mathrm{df.}}\ (\exists m_1\leq s)(\exists m_2\leq s)(\exists m_3\leq s)(\mathrm{Seql}(s,\mathbf{0}^{(4)})\wedge \mathrm{Term}(m_1)\wedge \mathrm{Term}(m_2)\wedge \mathrm{Term}(m_3)\wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(3)},\mathbf{0}^{(2)}]]]\in s\wedge [\mathbf{0}^{(2)},m_1]\in s\wedge [\mathbf{0}^{(3)},m_2]\in s\wedge [\mathbf{0}^{(4)},m_3]\in s).$$

Then the formula Atfmla(s) is true of a number if it codes an atomic formula:

$$\mathrm{Atfmla}(s)=_{\mathrm{df.}}\ \mathrm{Atf}^{=}(s)\vee \mathrm{Atf}^{A}(s)\vee \mathrm{Atf}^{M}(s).$$

Lecture VI

Truth and Satisfaction in RE (Continued).

We are getting closer to specifying the basis set. This set (let's call it B) will include the set of numbers [a,s] where a is a code of an atomic formula A and s is a function that is sufficient for A and satisfies A. B will include other (numbers coding) pairs of formulae and functions as well, as we will see later, but we can now start the construction of the formula that defines B by exhibiting the disjuncts of that formula that "correspond" to the cases in which the number a in [a,s] codes an atomic formula.

The first disjunct will be a formula true of two numbers a,s if a is an atomic formula of the form $P^2_1t_1t_2$ and s is sufficient for and satisfies a:

$$D_1(a,s)=_{df.} (\exists m_1\leq a)(\exists m_2\leq a)(\mathrm{Seql}(a,\mathbf{0}^{(3)}) \wedge \mathrm{Term}(m_1) \wedge \mathrm{Term}(m_2) \wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(2)},\mathbf{0}^{(1)}]]]\in a \wedge [\mathbf{0}^{(2)},m_1]\in a \wedge [\mathbf{0}^{(3)},m_2]\in a \wedge (\exists y_1\leq a)(\exists y_2\leq a)(\mathrm{Den}(m_1,y_1,s) \wedge \mathrm{Den}(m_2,y_2,s) \wedge y_1=y_2).$$

Notice that we use the identity predicate of RE to define the relation of satisfaction restricted to codes of equalities and functions that satisfy them. Below, the predicates of addition and multiplication of RE are used analogously, and so will be the connectives and quantifiers in our definitions of the generating relations for complex formulae. This procedure for defining satisfaction, and hence truth, was first used by Tarski. In the case of a sentence, like **0**=**0**, Tarski's definition of truth comes down to the biconditional: **0**=**0** is true iff 0=0. Tarski's definition appeared when some logical positivists had expressed doubts about the possibility of a scientifically acceptable definition or theory of truth. Tarski showed that this way of defining satisfaction and hence truth existed, and that it had important uses in logic and mathematics.

The second disjunct will be a formula true of two numbers a,s if a is an atomic formula of the form $P^3_1t_1t_2t_3$, where t_1, t_2 and t_3 are terms of RE, and s is sufficient for and satisfies a:

$$D_2(a,s)=_{df.} (\exists m_1\leq a)(\exists m_2\leq a)(\exists m_3\leq a)(\mathrm{Seql}(s,\mathbf{0}^{(4)}) \wedge \mathrm{Term}(m_1) \wedge \mathrm{Term}(m_2) \wedge \mathrm{Term}(m_3) \wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(3)},\mathbf{0}^{(1)}]]]\in s \wedge [\mathbf{0}^{(2)},m_1]\in s \wedge [\mathbf{0}^{(3)},m_2]\in s \wedge [\mathbf{0}^{(4)},m_3]\in s \wedge (\exists y_1\leq a)(\exists y_2\leq a)(\exists y_3\leq a)(\mathrm{Den}(m_1,y_1,s) \wedge \mathrm{Den}(m_2,y_2,s) \wedge \mathrm{Den}(m_3,y_3,s) \wedge A(y_1,y_2,y_3)).$$

The third disjunct will be a formula true of two numbers a,s if a is an atomic formula of the form $P^3_2t_1t_2t_3$, where t_1, t_2 and t_3 are terms of RE, and s is sufficient for and satisfies a:

$$D_3(a,s)=_{df.} (\exists m_1\leq s)(\exists m_2\leq s)(\exists m_3\leq s)(\mathrm{Seql}(s,\mathbf{0}^{(4)}) \wedge \mathrm{Term}(m_1) \wedge \mathrm{Term}(m_2) \wedge$$

$$\mathrm{Term}(m_3) \wedge [\mathbf{0}^{(1)},[\mathbf{0}^{(5)},[\mathbf{0}^{(3)},\mathbf{0}^{(2)}]]]\in s \wedge [\mathbf{0}^{(2)},m_1]\in s \wedge [\mathbf{0}^{(3)},m_2]\in s \wedge [\mathbf{0}^{(4)},m_3]\in s \wedge$$
$$(\exists y_1\le a)(\exists y_2\le a)(\exists y_3\le a)(\mathrm{Den}(m_1,y_1,s) \wedge \mathrm{Den}(m_2,y_2,s) \wedge \mathrm{Den}(m_3,y_3,s) \wedge M(y_1,y_2,y_3)).$$

Besides these three kinds of pairs [a,s], the basis set B will contain a fourth kind of pairs. The reason for this is that we are trying to avoid having to define the set of formulae of RE and the relation of a function being sufficient for a formula of RE. We are going to give generating relations that will generate the set {[a,s]: Sat(a,s)} in one fell swoop, as we said. But in order to do this, we need some way of dealing with the natural generating rule for disjunctions: if s satisfies A, B is a formula and s is sufficient for B, then s satisfies $(A\vee B)$. How are we going to define the relation corresponding to this rule without having defined the notions of being a formula and of being a function sufficient for a formula? We can appeal only to the notion of satisfaction as holding between less complex formulae and functions sufficient for them. Clearly, defining the relation corresponding to the following rule will not do: if s satisfies A and s satisfies B, then s satisfies $(A\vee B)$. It will not do because B may be unsatisfiable, in which case $(A\vee B)$ will not be generated.

To get around this we will use the following observation. All formulae of the form $(x_i<\mathbf{0})B$ are satisfied by all functions sufficient for them, since there is no number less than 0, so all pairs [a,s] where a is a formula of that form and s is sufficient for B must be in our final set. But if we have already generated the pairs consisting of a formula of the form $(x_i<\mathbf{0})B$ and all the functions s that satisfy it, (which we have to do in any case), then it must be the case that both B is a formula and s is sufficient for it. So the rule: if s satisfies A and s satisfies $(x_i<\mathbf{0})B$ then s satisfies $(A\vee B)$, will be appropriate to generate all the pairs of disjunctions and sequences that satisfy them, provided we have taken care of generating all the pairs [a,s] where a is a formula of the form $(x_i<\mathbf{0})B$ and s is sufficient for B. In fact we will take care of generating all the pairs [a,s] where a is a formula of the form $(x_i<t)B$, t is a term of RE not containing x_i, s is sufficient for $(x_i<t)B$ and the denotation of t with respect to s is 0 (for the same reason as above, all these pairs are in the relation of satisfaction).This is the reason for having a fourth kind of pairs in the basis set B: they are the pairs [a,s] where a is a formula of the form $(x_i<t)B$, B is atomic, s is sufficient for $(x_i<t)B$ and the denotation of t with respect to s is 0.

In order to give a formula that defines this relation, let's introduce the following formulae:

$$s(i)=m =_{\mathrm{df.}} [i,m]\in s;$$

this is simply a convenient abbreviation. The formula OcAtfmla(v,a) is true of v,a if v codes a variable, a is an atomic formula and the variable coded by v appears in the formula coded by a (notice that, in this case, the variable coded by v must appear free, since a has no quantifiers):

$$\mathrm{OcAtfmla}(v,a) =_{\mathrm{df.}} \mathrm{Atfmla}(a) \wedge \mathrm{Vbl}(v) \wedge (\exists m \leq a)(\exists i \leq a)(\mathrm{Term}(m) \wedge [i,m] \in a \wedge (\exists j \leq m)([j,v] \in m)).$$

We now give a formula that defines the useful relation of concatenation between sequences: Cxyz will be true of m,n,p if they code sequences and p codes the sequence that results from concatenating the sequence coded by m and the sequence coded by n, in this order:

$$\mathrm{C}xyz =_{\mathrm{df.}} (\exists m \leq x)(\exists n \leq y)(\mathrm{Seql}(x,m) \wedge \mathrm{Seql}(y,n) \wedge \mathrm{Seql}(z, m+n) \wedge x \subseteq z \wedge (j \leq n)(w \leq y)([j,w] \in y \supset [j+n,w] \in z)).$$

We can naturally define iterations of this relation, e.g.,

$$\mathrm{C}^3wxyz =_{\mathrm{df.}} (\exists m \leq z)(\mathrm{C}wxm \wedge \mathrm{C}myz).$$

Now we give a formula $D_4(a,s)$ which is true of a,s if a is a formula of the form $(x_i<t)B$ (where, therefore, x_i does not occur in t), B is atomic, s is sufficient for $(x_i<t)B$ and the denotation of t with respect to s is 0:

$$\begin{gathered} D_4(a,s) =_{\mathrm{df.}} (\exists m_1 \leq a)(\exists m \leq a)(\exists v \leq a)[\mathrm{Term}(m) \wedge \mathrm{Term}(m_1) \wedge \mathrm{Vbl}(v) \wedge [\mathbf{0}^{(1)},v] \in m_1 \wedge \\ (j \leq m)(w \leq m)([j,w] \in m \supset w \neq v) \wedge \\ (\exists b_1 \leq a)(\exists b_2 \leq a)(\mathrm{C}b_1b_2a \wedge \mathrm{Atfmla}(b_2) \wedge \mathrm{Seql}(b_1,\mathbf{0}^{(5)}) \wedge b_1(\mathbf{0}^{(1)})=[\mathbf{0},\mathbf{0}] \wedge b_1(\mathbf{0}^{(2)})=m_1 \wedge \\ b_1(\mathbf{0}^{(3)})=[\mathbf{0},\mathbf{0}^{(3)}] \wedge b_1(\mathbf{0}^{(4)})=m \wedge b_1(\mathbf{0}^{(5)})=[\mathbf{0},\mathbf{0}^{(1)}]) \wedge \\ (v_1 \leq a)((\mathrm{Vbl}(v_1) \wedge v \neq v_1 \wedge (\exists j \leq a)([j,v_1] \in m)) \vee \mathrm{OcAtfmla}(v_1,a)) \supset (\exists k \leq a)([v_1,k] \in s)) \wedge \\ \mathrm{Den}(m,0,s)]. \end{gathered}$$

Finally, our basis set B is defined by the following formula of Re:

$$\mathrm{Basis}(x) =_{\mathrm{df.}} (\exists a \leq x)(\exists s \leq x)(x=[a,s] \wedge (D_1(a,s) \vee D_2(a,s) \vee D_3(a,s) \vee D_4(a,s))).$$

Hence, B is r.e.

We turn now to defining the relations that will generate the set G from the basis set B. These will correspond fairly closely to the clauses in the inductive definition of satisfaction for assignments and formulae of RE. But we also have to take care of generating the more and more complex formulae of the form $(x_i<t)B$ where s is sufficient for $(x_i<t)B$ and the denotation of t with respect to s is 0.

We will define first the relations corresponding to the clauses in the inductive definition of satisfaction. First we consider the rule corresponding to the clause for conjunction: if s satisfies A and s satisfies B, then s satisfies $(A \wedge B)$, or schematically:

$$\frac{\text{Sat (A, s), Sat (B,s)}}{\text{Sat ((A} \wedge \text{B), s).}}$$

All we need to define an appropriate generating relation is a formula that defines the relation {<<A, s>, <B, s>, <(A $\wedge$ B),s>>: A, B are sequences}; or, to be exact, a formula that defines the relation R_C={<[m,s], [n,s], [p,s]>: p codes the conjunction of the sequences that m and n code} (notice that if two pairs [a,s], [b,s] have been already generated, there is no further need to require that a and b be formulae and s be a finite function sufficient for a and b). Now, the formula Conj(x,y,z) is true of m,n,p if p is the conjunction of m and n:

$$\text{Conj}(x,y,z)=_{\text{df.}} (\exists l\leq z)(\exists c\leq z)(\exists r\leq z)(\text{Seql}(w,\mathbf{0}^{(1)}) \wedge \text{Seql}(c,\mathbf{0}^{(1)}) \wedge \text{Seql}(r,\mathbf{0}^{(1)})\wedge l(\mathbf{0}^{(1)})=[\mathbf{0},\mathbf{0}] \wedge c(\mathbf{0}^{(1)})=[\mathbf{0},\mathbf{0}^{(6)}] \wedge r(\mathbf{0}^{(1)})=[\mathbf{0},\mathbf{0}^{(1)}] \wedge C^5lxcyrz).$$

Given this, the RE formula with free variables x,y,z

$$(\exists s\leq x)(\exists m\leq x)(\exists n\leq y)(\exists p\leq z)(x=[m,s] \wedge y=[n,s] \wedge z=[p,s] \wedge \text{Conj }(m,n,p))$$

defines the generating relation we need.

For disjunction, we have two rules; schematically:

$$\frac{\text{Sat (A,s), B is a formula of RE and s is sufficient for B}}{\text{Sat ((A} \vee \text{B),s)}}$$

and

$$\frac{\text{Sat (B,s), A is a formula of RE and s is sufficient for A}}{\text{Sat ((A} \vee \text{B),s).}}$$

We could define the corresponding relation in a way analogous to the case of disjunction if we had defined the notion of formula of RE and the notion of being a function sufficient for a formula, but this is what we set ourselves to avoid. It is here that we will appeal to the trick explained above. For example, the first rule becomes:

$$\frac{\text{Sat}(A,s),\ \text{Sat}((x_i<\mathbf{0})B,s)}{\text{Sat}((A \vee B),s).}$$

Now we need a formula that defines the relation $\{<<A, s>, <(x_i<\mathbf{0})B, s>, <(A \vee B),s>>: A, B$ are sequences$\}$; or, to be exact, a formula that defines the relation $R_{D1}=\{<[r,s], [n,s], [p,s]>:$ n is a concatenation of $(x_i<\mathbf{0})$ (for some i) and some sequence q, and p codes the disjunction of the sequences that r and q code$\}$. Now, a formula Disj(x,y,z) true of r,n,p if p is the disjunction of m and n is easily definable as in the case of conjunction. Assuming this, the RE formula with free variables x,y,z

$$\begin{gathered}(\exists s\leq x)(\exists r\leq x)(\exists n\leq y)(\exists q\leq y)(\exists p\leq z)(x=[r,s] \wedge y=[n,s] \wedge z=[p,s] \wedge \\ (\exists m_1\leq n)(\exists v\leq n)[\text{Term}(m_1) \wedge \text{Vbl}(v) \wedge [\mathbf{0}^{(1)},v]\in m_1 \wedge \\ (\exists b\leq n)(\text{Cbqn} \wedge \text{Seql}(b,\mathbf{0}^{(5)}) \wedge b(\mathbf{0}^{(1)})=[\mathbf{0},\mathbf{0}] \wedge b(\mathbf{0}^{(2)})=m_1 \wedge b(\mathbf{0}^{(3)})=[\mathbf{0},\mathbf{0}^{(3)}] \wedge \\ b(\mathbf{0}^{(4)})=[\mathbf{0}^{(2)},\mathbf{0}^{(1)}] \wedge b(\mathbf{0}^{(5)})=[\mathbf{0},\mathbf{0}^{(1)}]] \wedge \\ \text{Disj}(r,q,p))\end{gathered}$$

defines the generating relation we need. The second rule for disjunction corresponds to another generating relation, R_{D2}, that can be defined analogously.

The rule for existential quantification is a bit more complicated:

$$\frac{\text{Sat}(A,s),\ s_1 \text{ differs from } s \text{ at most in what it assigns to } x_i}{\text{Sat}((\exists x_i)A,s_1)}$$

In order to define the appropriate generating relation we need a formula that defines the relation $\{<<A, s>, <(\exists x_i)A, s_1>>:$ A is a sequence and s_1 differs from s (if at all) in what it assigns to $x_i\}$; or, to be exact, a formula that defines the relation $R_{EQ}=\{<[m,s], [n,s_1]>:$ n codes the concatenation of $(\exists x_i)$ (for some i) and m, and s_1 differs from s (if at all) in what it assigns to $x_i\}$. Now, a formula ExQu(m,y,p) true of x,i,z if p is a Gödel number of the concatenation of $(\exists x_i)$ and m, is easily definable as in the above cases (remembering our special way of coding terms when they appear in formulae). We also need to have an RE formula Diff(s,s_1,v) that says that s and s_1 assign the same to variables other than v. This can be done as follows:

$$\begin{gathered}\text{Diff}(s,s_1,v) =_{\text{df.}} \text{Funct}(s) \wedge \text{Funct}(s_1) \wedge \text{Vbl}(v) \wedge (w\leq s)(p\leq s)(q\leq s_1)((\text{Vbl}(w) \wedge w\neq v \wedge \\ [w,p]\in s \wedge [w,q]\in s_1)\supset p=q).\end{gathered}$$

Then R_{EQ} is definable by the following formula of RE with free variables x,y:

$$(\exists s\leq x)(\exists s_1\leq y)(\exists i\leq y)(\exists m\leq y)(\exists p\leq y)(x=[m,s] \wedge \mathrm{ExQu}(m,i,p) \wedge y=[p,s_1] \wedge \mathrm{Diff}(s,s_1,i)).$$

Finally, we come to bounded universal quantification, which is a bit subtler than the preceding cases. We have the rule

$$\frac{\text{Sat}((x_i<\mathbf{0}^{(n)})A,s),\ \text{Sat}(A,s_1),\ s_1 \text{ differs from } s \text{ only in that } s_1 \text{ assigns } n \text{ to } x_i,\ t \text{ denotes } n+1 \text{ with respect to } s,\ x_i \text{ does not occur in } t}{\text{Sat}((x_i < t)A,s)}$$

This says, roughly, that if s satisfies both $(x_i<\mathbf{0}^{(n)})A$ and $A(\mathbf{0}^{(n)})$, then s satisfies any formula of the form $(x_i<t)A$ where t denotes n+1 with respect to s. The corresponding relation is R_{UQ}={<[m,s], [n,s_1], [p,s]>: for some term t and some variable x_i not occurring in t, p codes the concatenation of $(x_i < t)$ and n, s_1 differs from s at most in that it assigns a number q to x_i, and m is the concatenation of $(x_i<\mathbf{0}^{(q)})$ and n}. Now, a formula UnQu(m,i,t,p) which “says” that p is the Gödel number of $(x_i<t)A$, where A is the formula whose Gödel number is m, is easily definable as in the above cases (taking care of remembering that we are not using the simple-minded coding scheme). Then we can define R_{UQ} by means of the following formula with free variables x,y,z:

$$(\exists s\leq x)(\exists s_1\leq y)(\exists m\leq x)(\exists n\leq y)(\exists p\leq z)(\exists i\leq x)(\exists t\leq z)(\exists q\leq x)(\exists r\leq x)(\mathrm{Term}(t) \wedge \mathrm{Vbl}(i) \wedge (j\leq t)(w\leq t)([j,w]\in t\supset w\neq i) \wedge \mathrm{UnQu}(n,i,t,p) \wedge \mathrm{Diff}(s,s_1,i) \wedge \mathrm{Num}(q,r) \wedge \mathrm{Den}(i,r,s_1) \wedge \mathrm{UnQu}(n,i,q,m) \wedge x=[m,s] \wedge y=[n,s_1] \wedge z=[p,s]).$$

We are now done in our job of defining the generating relations corresponding to the clauses in the inductive definition of satisfaction for RE. We now have to make sure that we can define the relations that generate more and more complex formulae of the form $(x_i<t)A$.

For conjunction, we have the rule

$$\frac{\text{Sat}((x_i<t)A, s),\ \text{Sat}((x_i<t)B,s),\ t \text{ denotes } 0 \text{ with respect to } s}{\text{Sat}((x_i<t)(A \wedge B), s).}$$

The corresponding relation is R_{C*}={<[m,s], [n,s], [p,s]>: for some q,r,x_i,t such that x_i is a variable, t is a term, m codes the concatenation of $(x_i<t)$ and q, n codes the concatenation of $(x_i<t)$ and r, the denotation of t with respect to s is 0 and p is the concatenation of $(x_i<t)$ and the conjunction of q and r}. R_{C*} is definable by the following formula with free variables x,y,z:

$$(\exists s\leq x)(\exists m\leq x)(\exists n\leq y)(\exists p\leq z)(\exists q\leq x)(\exists r\leq y)(\exists i\leq m)(\exists t\leq m)(x=[m,s] \wedge y=[n,s] \wedge z=[p,s] \wedge \mathrm{Vbl}(i) \wedge \mathrm{Term}(t) \wedge \mathrm{UnQu}(q,i,t,m) \wedge \mathrm{UnQu}(r,i,t,n) \wedge \mathrm{Den}(t,\mathbf{0},s) \wedge (\exists w\leq p)(\mathrm{Conj}(q,r,w) \wedge \mathrm{UnQu}(w,i,t,p))).$$

For disjunction, we have the rule

$$\frac{\text{Sat }((x_i<t)A,\ s),\ \text{Sat }((x_i<t)B,s),\ t \text{ denotes } 0 \text{ with respect to } s}{\text{Sat }((x_i<t)(A \vee B),\ s),}$$

whose corresponding generating relation R_{D*} is definable similarly.

For existential quantification we have the rule

$$\frac{\text{Sat }((x_i<t)A,\ s_1),\ t \text{ denotes } 0 \text{ with respect to } s, \text{ and } s_1 \text{ differs from } s \text{ at most in what it assigns to } x_j}{\text{Sat }((x_i<t)(\exists x_j)A,\ s),}$$

whose corresponding relation R_{EQ*} is easily definable by means of the formulae that we already have.

The same is true for the relation R_{UQ*} corresponding to the relevant rule for bounded universal quantification:

$$\frac{\text{Sat }((x_i<t)A,\ s_1),\ t \text{ denotes } 0 \text{ with respect to } s, \text{ and } s_1 \text{ differs from } s \text{ at most in what it assigns to } x_j,\ t_1 \text{ is a term and } x_j \text{ is a variable not appearing in } t_1}{\text{Sat }((x_i<t)(x_j<t_1)A,\ s).}$$

Thus we finish our specification of the generating relations. It can be proved (by induction on the complexity of the formulae) that for every formula A and function s, if a codes A and s is a function that satisfies A, then [a,s] is generated from B by the relations R_C, R_{D1}, R_{D2}, R_{EQ}, R_{UQ}, R_{C*}, R_{D*}, R_{EQ*}, R_{UQ*}. So G is indeed the set generated from B, which is r.e., by these relations, which are r.e. Using the Generated Sets Theorem, we then reach the result that G is r.e., which is what we had set out to prove.

Given that RE contains its own satisfaction predicate, it also contains its own truth predicate, that is, there is a formula of RE Tr(m) with one free variable which is true of a (Gödel number of a) formula if and only if the formula is true in the intended interpretation of RE. Noting that a formula is true iff it is satisfied by a Gödel number of the empty sequence, we can define

$$\mathrm{Tr}(m) =_{\mathrm{df.}} \mathrm{Sat}(m,\mathbf{0}^{(3)}).$$

(3 codes the empty sequence, for it is not in the range of our pairing function.)

If we had shown that RE contains its own truth predicate Tr(x), we could then have defined satisfaction with its help, using a remark due to Tarski: a number m satisfies a formula $A(x_1)$ with one free variable just in case the sentence $(\exists x_1)(x_1 = \mathbf{0}^{(m)} \wedge A(x_1))$ is true, so easily, since the concatenation function is definable in RE, we can define satisfaction in terms of truth within RE.

<u>Exercises</u>

1. Define the *characteristic function* of a set S to be the function which, for every natural number x, takes value 1 if $x \in S$, and value 0 if $x \notin S$. Similarly, the characteristic function of an n-place relation R is the function that for every n-tuple $<x_1,...,x_n>$, takes value 1 if $<x_1,...,x_n> \in R$, and value 0 if $<x_1,...,x_n> \notin R$. The *weak characteristic function* of a set S (or n-place relation R) takes value 1 on a number x (or n-tuple $<x_1,...,x_n>$) if x ($<x_1,...,x_n>$) belongs to S (R). (a) Show that a set or relation is recursive iff its characteristic function is. (b) Show that a set or relation is r.e. iff its weak characteristic function is partial recursive. (c) Show that the range and domain of any recursive function in one variable is r.e.

2. Show that the relation $z=x^y$ is r.e., and therefore that the exponentiation function is recursive by both the method of primitive recursion and the method of generated sets. How do the two defining formulae in RE differ from each other?

3. Use the Generated Sets Theorem to show that the Ackermann function is recursive. Where would an argument that the Ackermann function is primitive recursive break down?

4. Show that the set of Gödel numbers of formulae of the first order language of arithmetic is r.e., using the Generated Sets Theorem. Do the problem explicitly as follows. Write the basis clauses and generating clauses for the formulae themselves, and also show how they are then translated into basis clauses and generating clauses for codes. Then indicate what formula of RE results if we apply the argument in the proof of the Generated Sets Theorem to this case. Do the same for the set of formulae obeying the nested quantifier restriction.

Lecture VII

The Enumeration Theorem. A Recursively Enumerable Set which is Not Recursive

Now that we have shown how to define satisfaction for RE within RE itself, we are ready to prove what may be considered the fundamental theorem of recursion theory. First, let us define a variant on the notion of satisfaction we defined above. If B is a formula whose only free variable is x_1, then we say the number n satisfies$_1$ B just in case the unit sequence <n> satisfies B. We may thus define

$$\text{Sat}_1(m, n) =_{\text{df.}} (\exists s)(\text{Sat}(m, s) \land (y\in s)(y=[[\mathbf{0}^{(1)},\mathbf{0}^{(1)}],n]) \land [[\mathbf{0}^{(1)},\mathbf{0}^{(1)}],n]\in s).$$

One could in general define Sat_k, satisfaction for formulae with k free variables, either directly in a similar manner or by using the k-tupling function to reduce it to the case k=1.

We also use the notation W(e,n) for the relation that holds just in case $\text{Sat}_1(e,n)$ holds. We now have the

Enumeration Theorem: There is a 2-place r.e. relation W such that for every r.e. set S, there is an e such that S = {n: W(e,n)}.

Proof: Let W_e={x: W(e,x)}. Each set W_e is defined by $\text{Sat}_1(\mathbf{0}^{(e)},x)$ and is therefore r.e. If, on the other hand, S is an r.e. set, then it is defined by some RE formula B with one free variable. We may assume that B's free variable is x_1; letting e be a Gödel number of B, we see that S = W_e.

The Enumeration Theorem is so called because W enumerates the r.e. sets. This theorem is a standard theorem of recursion theory, though our presentation of it is not standard. When recursion theory is presented in terms of Turing machines, for example, W(e, x) is usually the relation *e codes a Turing machine which gives output "yes" on input x* for some fixed method of coding up Turing machines, and is shown to be r.e. by constructing a Turing machine which decodes the instructions given in e and applies them to the input x. In each formalism for developing recursion theory, the relation W(e, x) will be a different relation. (The notation 'W' originates in Kleene.)

In general, we can define a k+1-place relation W^{k+1} which holds of a k+1-tuple $<e,n_1,...,n_k>$ if W(e, $[n_1,...,n_k]$) holds. This can be used to prove that W^{k+1} enumerates the k-place r.e. relations.

A very famous corollary of the Enumeration Theorem is the following:

Theorem. There is an r.e. set whose complement is not r.e. (thus, an r.e. set which is not recursive).
Proof: Let K be the set {x: W(x, x)}. K is clearly r.e., since it is defined by the formula $Sat_1(x_1, x_1)$. However, -K is not r.e., and so K is not recursive. To see this, suppose -K were r.e. Then we would have $-K = W_e$ for some e. By the definition of K, we see that $x \in -K$ iff $x \notin W_x$, so in particular $e \in -K$ iff $e \notin W_e$. But $W_e = -K$, so $e \in -K$ iff $e \notin -K$, contradiction.

(This shows that negation is not definable in RE: if it were, the complement of any r.e. set would be definable in RE, so in particular -K would be definable in RE.) This proof uses an idea due to Cantor. As we will see, once we have this theorem, Gödel's first incompleteness theorem is just around the corner.

The fact (if it is one) about the intuitive notion of computability corresponding to the enumeration theorem is that there is a semi-computable relation that enumerates all the semi-computable sets. It follows from this that there is a semi-computable set that is not computable. However, to prove the enumeration theorem for semi-computability, and thus to prove that not all semi-computable sets are computable, it seems necessary to use Church's Thesis. If there were a single language in which all computation procedures could be written out, then the enumeration theorem would follow: simply find some way of coding up this language numerically, and some effective way of decoding coded instructions and applying them to arguments. However, if we do not assume Church's Thesis, then it is by no means obvious that there is such a language. At first glance it might appear that the lesson of Gödel's work is that there is no single language in which all computation can be represented, just as there is no single fully classical language in which everything can be expressed. Every language will have some sort of Gödel-type limitation; it is a peculiarity of the language RE that the limitation is not that it cannot express its own semantic notions (as is the case with full first-order languages), but that it cannot express negation. But if it turns out that there are some semi-computable sets and relations that are not expressible in RE, then it is quite conceivable that all semi-computable sets and relations are computable and that the enumeration theorem for semi-computability fails.

The fact that the enumeration theorem is so fundamental to recursion theory, and that its proof for semi-computability requires Church's Thesis, indicates a limitation to how much recursion theory can be developed for the informal notion of computability by starting with intuitively true axioms about computability. Shoenfield tries this approach in his book on recursion theory; he winds up assuming the enumeration theorem, and does not give a fully convincing intuitive justification for it, since he in effect assumes that there is a single language in which all computation procedures can be represented.

The Road from the Inconsistency of the Unrestricted Comprehension Principle to the Gödel-Tarski Theorems

The result that RE contains its own truth and satisfaction predicates may seem surprising, since it is commonly said that Gödel and Tarski showed that no language can contain its own truth or satisfaction predicates. This is true for a wide range of languages, but the language RE is not among them. We shall now look at their result and at is historical roots.

Early on in the 20th century, it was discovered, to the surprise of many, that a certain set-theoretic principle is self-contradictory. The principle is called the *unrestricted comprehension scheme*:

$$(z_1)...(z_n)(\exists y)(x)(x \in y \equiv A(x, z_1, ..., z_n))$$

where A is any formula in the language of set theory whose free variables are among x and $z_1, ..., z_n$ (in particular, A does not contain y free). $z_1, ..., z_n$ are called *parameters*. In the case n = 0, we have the *parameter-free* unrestricted comprehension scheme:

$$(\exists y)(x)(x \in y \equiv A(x))$$

It is important to note that A may itself contain the predicate '$\in$'.

Russell showed that the unrestricted comprehension scheme, even in its parameter-free version, is self-contradictory: simply take A(x) to be the formula $\sim x \in x$. We then have

$$(x)(x \in y \equiv \sim x \in x)$$

for some y, from which it follows that

$$y \in y \equiv \sim y \in y$$

which is directly self-contradictory. This observation is called *Russell's paradox*.

Russell got the idea of his paradox by analyzing Cantor's proof via diagonalization that there is no function mapping a set onto its powerset, and applying it to a more complicated paradox that embedded his. The Russell paradox is not the only set-theoretic paradox. Other paradoxes were discovered at the very time of the formation of set theory itself. For example, there is the Burali-Forti paradox, and the paradox of the greatest cardinal. In general, these paradoxes, like the Russell paradox, can be used to show that the unrestricted comprehension scheme is inconsistent, or at least, that it leads to an inconsistency in conjunction with the axiom of extensionality.

If the unrestricted comprehension scheme is logically self-contradictory, this cannot depend in any way on the interpretation of '$\in$'. From a purely formal point of view, it doesn't matter whether '$\in$' means 'is a member of' or 'is satisfied by'. In fact, if you knew somehow that the unrestricted comprehension scheme is inconsistent but didn't know why, you would be able to see immediately that no first-order language can contain its own satisfaction predicate. Consider the following interpretation of a first-order language with '$\in$' as its only predicate: let the variables range over the formulae with one free variable of the language, and assume that the language contains its own expressions in its own domain. Suppose that we interpret '$\in$' by means of the 2-place relation S(y,x) which holds just in case y is a formula with one free variable and x satisfies y. Then suppose that the language could define its own satisfaction for formulae of one free variable. Then we would have a model that would make the comprehension scheme true, because a formula A(x) can be taken to be the object y, and then the principle says that x satisfies y if and only if A(x), which is of course true. So any proof of the inconsistency of the comprehension principle proves that a first order language cannot contain its own satisfaction predicate for formulae with one free variable. Often, the results of Gödel and Tarski are presented as if they involved all kinds of sophisticated ideas very different from these, but the inconsistency of the unrestricted comprehension scheme is essentially what proves those results.

(Now, suppose that in the derivation of a paradox we used the unrestricted comprehension axiom with n parameters. How are we going to interpret '$x \in y$' in this case? One way will be analogous to the reduction of satisfaction to truth that we saw for the case of RE. We take y to range over the formulae with one free variable, or their codes, and '$x \in y$' to be defined by "x satisfies y", where y will be, or code, the formula with one free variable

$$(\exists z_1)...(\exists z_n)(z_1=\mathbf{0}^{(m_1)} \wedge ... \wedge z_n=\mathbf{0}^{(m_n)} \wedge A(x, z_1, ..., z_n)),$$

where $m_1,...,m_n$ are the parameters. We could also use a relation of substitution of terms for free variables in formulae to specify how y will be, in a way indicated below. Or we could define '$x \in y$' with the help of the pairing function, as holding between a number and a pair composed of (the code of) the formula $A(x, z_1, ..., z_n)$ and a finite function assigning values to all of the variables $z_1, ..., z_n$.)

We can use what we have learned about satisfaction to state some very general results about the indefinability of truth. As long as the language L contains a name 'a' for each object a in its domain, we can (in the metalanguage) define satisfaction for L in terms of truth for L: an object a satisfies a formula $A(x_1)$ just in case the sentence A('a') is true (where A('a') is got from $A(x_1)$ by replacing all free occurrences of x_1 with 'a'). We can turn this into a definition in L of satisfaction in terms of truth as long as L possesses certain syntactic notions. Suppose, for example, L has function symbols Q and S denoting functions q and s, where q(a) = 'a' for all a in L's domain, and $s(A(x_1), t) = A(t)$ for all formulae $A(x_1)$ and terms t. Then $s(A(x_1), q(a)) = A('a')$ for all $A(x_1)$ and a, and so the formula Tr(S(y,Q(x))) of L will define satisfaction in L if Tr defines truth in L. Since satisfaction for L is not definable in L, it follows that truth is not definable either. So in general we see that, for fully classical L, the following conditions cannot jointly obtain:

1. Truth for L is definable in L.
2. The relation of substitution of terms for free variables is definable in L.
3. Every object in L's domain has a name, and moreover the function from an object to its name is definable in L.

In fact, somewhat weaker conditions than 2 and 3 are also incompatible with 1. For example, 3 may be replaced with: every object in L's domain is denoted by some term of L and the relation *t denotes x* is definable in L. For suppose Den(t, x) defined that relation; then the formula $(\exists z)(\mathrm{Den}(z, x) \wedge \mathrm{Tr}(S(y, z)))$ would define satisfaction in L in terms of truth. (We could further weaken 3 by assuming only that every object a of L's domain has a definite description D in L, and that we can specify D in terms of a within L.) Also, we can use the trick remarked on by Tarski to avoid using the substitution function. An object a will satisfy $A(x_1)$ just in case the sentence $(\exists x_1)(x_1 = \text{'a'} \wedge A(x_1))$ is true (or $(\exists x_1)(x_1 = t \wedge A(x_1))$ is true for some term t denoting a, or $(\exists x_1)(D(x_1) \wedge A(x_1)$ is true for some definite description D of a), so as long as the function $F(t, A(x_1)) = (\exists x_1)(x_1 = t \wedge A(x_1))$ is definable in L, and 3 obtains, we can define satisfaction in terms of truth within L. Moreover, F is itself easily definable in terms of concatenation (as long as terms for the primitive symbols of L exist in L), and is anyway simpler than the substitution function, which has to distinguish between free and bound occurrences of variables. To put it in a succinct form, we see that a language cannot both define all the devices that can be used to reduce truth to satisfaction and contain its own truth predicate.

So far, we have been concentrating on interpreted languages whose domains include all the expressions of the language itself. However, we can generalize the discussion by considering languages that can talk about their own expressions indirectly, via coding. Let L be a language with a countable vocabulary and with an infinite domain D. Suppose we had a function f mapping the elements of some subset D_1 of D onto the formulae, or at least onto the formulae with one free variable. Call this a coding function. Given any coding function f for L, the relation $\{<y,x>: x \text{ is in } D_1 \text{ and } f(y) \text{ is satisfied by } x\}$ of "coded satisfaction" between elements of L's domain is not itself definable in L: if S(y,x) defined it, then the unrestricted comprehension scheme, with '$x \in y$' replaced by 'S(y,x)', would be true in L, which we know to be impossible. Note that none of this depends on any particular facts about f; any coding function will do, and we know that such a mapping will exist whenever L has an infinite domain.

As before, this is related to the question of the definability of truth in L. Let us say that a formula Tr(x) of L defines truth in L (relative to f) if Tr(x) defines {y: f(y) is true in L}. We can always find a function f such that some formula of L defines truth in L relative to f (for practically any L). For example, let L be the language of arithmetic, and let f "assign"

Gödel numbers to sentences of L so that the Gödel numbers of the true sentences of L are 0, 2, 4, ... and the Gödel numbers of false sentences of L are 1, 3, 5, ... (Or if arbitrary formulae are to be coded rather than just sentences, we can let 0, 3, 6, ... code the true sentences, let 1, 4, 7, ... code the false sentences, and let 2, 5, 8, ... code the rest of the formulae.) Relative to such a Gödel numbering, truth in L is obviously definable in L. However, for any coding f, conditions 1-3 above will still not be jointly satisfiable, and so for this particular Gödel numbering, the basic syntactic notions will not be definable in L. For any ordinary Gödel numbering (e.g. our own numbering), the syntactic notions will be definable, and so truth will be undefinable. But again, satisfaction will be undefinable for any f, and for any L.

Now, as we have seen, if we drop the requirement that L be fully classical, these indefinability results will no longer hold, since the language RE has its own satisfaction predicate. However, RE does not have its own *unsatisfaction* predicate, i.e. there is no formula U(y,x) of RE that obtains just in case x fails to satisfy y. (If there were, then we could define -K in RE by U(x_1, x_1).) More generally, if L is a language with classical semantics, but not necessarily with all classical connectives, then unsatisfaction is not definable in L. To see this, suppose we had an unsatisfaction predicate U(y,x). Then letting u be the formula U(y,y), we would have that U(u,y) obtains iff y does not satisfy U(y,y), and so U(u, u) obtains iff u does not satisfy U(y,y); but to say that U(u, u) obtains is just to say that u satisfies U(y, y), so this is impossible. Similarly, given suitable restrictions on L, a language cannot have its own untruth predicate. Similar remarks apply when we allow languages to talk about their formulae indirectly via codes.

The enumeration theorem is really a form of the naive comprehension scheme for RE, since the content of the theorem is that for every RE formula A(x_1) there is an e such that

$$(x_1)(A(x_1) \equiv W(e, x_1))$$

We cannot derive a contradiction by letting A be the formula ~W(x_1, x_1), since negation is not definable in RE. This shows that it is essential to use either negation or unbounded universal quantification in showing that scheme to be inconsistent for classical languages, since RE lacks both negation and unbounded universal quantification. In fact, however, there are languages which have unbounded universal quantification, as well as the other logical symbols of RE, but which lack negation, and which have their own satisfaction predicates; so only the use of negation is really essential.

From all of this it follows that, given our Gödel numbering, the language of arithmetic does not have its own truth predicate. This was originally shown by Tarski as an application of the work of Gödel. However, there it was presented in a more complicated way as an application of the liar paradox. First, Gödel's self-reference theorem was used to obtain, for any formula T(x_1) of the language of arithmetic, a sentence A such that

$$A \equiv \sim T(\mathbf{0}^{(n)})$$

is true, where n is the Gödel number of A. If $T(x_1)$ defined the set of truths of arithmetic, then we would also have

$$A \equiv T(\mathbf{0}^{(n)})$$

from which $T(\mathbf{0}^{(n)}) \equiv \sim T(\mathbf{0}^{(n)})$ follows, which is self-contradictory. Intuitively, if $T(x_1)$ means "x_1 is true", then the sentence A says of itself that it is not true; so the Tarski-Gödel proof can be seen as an application of the liar paradox. However, the construction of the sentence A is rather tricky, and leaves one with the impression that something much more subtle is going on here than actually is. In fact, when one takes the Tarski-Gödel proof apart, one sees that it really boils down to the observation that arithmetic lacks its own satisfaction predicate, which in turn is a direct consequence of the Russell paradox, as we saw above.

Under the interpretation of '$\in$' as the relation of satisfaction, Russell's paradox is known as 'the paradox of 'heterological''. Grelling was the discoverer of this paradox, and he obtained it by reflecting on Russell's paradox. Call a predicate of English *autological* if it is true of itself, and *heterological* otherwise. (For example, 'polysyllabic' is autological and 'monosyllabic' is heterological.) A problem arises when we ask whether 'heterological' is itself heterological. 'heterological' is heterological iff 'heterological' is not true of 'heterological', iff 'heterological' is not heterological—a contradiction. If we interpret '$x \in y$' to mean 'y is true of x', then the formula '$x \notin x$' means that x is heterological, and the derivation of the Grelling paradox is formally identical to the above derivation of the Russell paradox.

Gödel mentions the liar paradox (and the paradox of Richard) as sources for the reasoning leading to his theorems. He does not mention Russell's paradox, or the paradox of 'heterological', although the Tarski-Gödel results are more naturally motivated by appeal to them, as we have seen. That Russell's paradox and Grelling's paradox can be most naturally put to this use is perhaps a fact known to some logicians, but to the author's knowledge, it is not mentioned in the printed literature.

In fact, some logicians have probably misunderstood the relation between the liar paradox on the one hand and Russell's and Grelling's paradoxes on the other. That the Gödel-Tarski results can be motivated in the two ways is no surprise, for, in fact, on one way of stating the liar paradox it just is the Grelling paradox. The liar paradox is traditionally stated in terms of sentences like 'This sentence is false'. One way to get a liar sentence without using locutions like 'this sentence' is via the sentence "Yields a falsehood when appended to its own quotation' yields a falsehood when appended to its own quotation'. Since the result of appending the phrase mentioned in the sentence to its own quotation is the sentence itself, the sentence says of itself that it is false. (A briefer way of writing the sentence is as follows: "Is not true of itself' is not true of itself'.) Quine gives this version of the liar paradox in "The Ways of Paradox". But he goes on to say that this antinomy is "on a par with the one about 'heterological'" ("The Ways of Paradox", in *The Ways of Paradox*, New York, Random House, 1966; p. 9). This is at best misleading, especially in this context, for the paradox simply *is* the Grelling paradox, since 'is heterological' means the same as 'yields a falsehood when appended to its own quotation'.

Lecture VIII

Many-one and One-one Reducibility.

Given that not all sets are recursive, and indeed that some r.e. sets are not recursive, we may want to ask of some nonrecursive set A whether the problem of deciding membership in it can be reduced to that of deciding membership in some other set B. This idea gives rise to a number of reducibility notions in recursion theory; in this section, we shall discuss two of the simplest such notions.

We say that a set A is *many-one reducible* to B (in symbols, $A \leq_m B$) if there is a total recursive function ϕ such that for all m, $m \in A$ iff $\phi(m) \in B$. (We also say *m-reducible* for *many-one reducible*.) We also write ϕ: $A \leq_m B$ when ϕ is a total recursive function satisfying this condition. If ϕ: $A \leq_m B$ and ϕ is 1-1, then we say that A is *1-1 reducible* (or *1-reducible*) to B (in symbols, $A \leq_1 B$).

One way to think of this informally is in terms of *oracles*. Suppose there were an oracle that could tell you, for an arbitrary number, whether it is an element of B. Then if $A \leq_m B$, you will have a way to use the oracle to find out whether an arbitrary number is an element of A: to see whether $m \in A$, simple compute $\phi(m)$ and consult the oracle. If the oracle tells you that $\phi(m) \in B$, then you know that $m \in A$, and if it tells you that $\phi(m) \notin B$, then you know that $m \notin A$.

Here's a simple example of 1-1 reducibility. Let A be an arbitrary set, and let B = {2m: $m \in A$}. Then we see that $A \leq_1 B$ by letting $\phi(m) = 2m$. And intuitively, we can effectively determine whether $m \in A$ by consulting the oracle about whether $2m \in B$.

It can readily be shown that the relations $\leq_1$ and $\leq_m$ are reflexive and transitive. Let us write $A \equiv_m B$ for $A \leq_m B$ & $B \leq_m A$, and similarly for $\equiv_1$; it then follows that $\equiv_m$ and $\equiv_1$ are equivalence relations. The $\equiv_m$-equivalence classes are called many-one degrees or m-degrees; similarly, the $\equiv_1$-equivalence classes are called 1-1 degrees or 1-degrees.

The relations $\leq_m$ and $\leq_1$ do not coincide. To see this, let A = {even numbers}, and let $\phi(m) = 0$ if m is even, 1 if m is odd. Then we see that ϕ: $A \leq_m \{0\}$. However, there is clearly no 1-1 function ϕ such that $x \in A$ iff $\phi(x) \in \{0\}$, so A is not 1-reducible to {0}.

A set is *many-one complete r.e.* (*1-1 complete r.e.*) iff it is r.e. and every r.e. set is many-one reducible (1-1 reducible) to it. If S is many-one complete r.e., then every r.e. set is of the form $\{x: \phi(x) \in S\}$ for some total recursive ϕ. One example of a many-one complete set (which is also 1-1 complete) is the set $S = \{[e, x]: x \in W_e\}$. To see that S is 1-1 complete, let S_1 be any r.e. set. S_1 is W_e for some e, so let $\phi(x) = [e, x]$; $x \in S_1$ iff $x \in W_e$ iff $\phi(x) \in S$. ϕ is clearly as required; in particular, ϕ is recursive, since its graph is defined by the formula $y = [\mathbf{0}^{(e)}, x]$. Another 1-1 complete set is the set T of Gödel numbers of true RE sentences. To see this, note that if $A(x_1)$ defines a set S_1, then $n \in S_1$ iff $A(\mathbf{0}^{(n)})$ is true; so if $\phi(n)$ is a recursive function that picks a Gödel number of $A(\mathbf{0}^{(n)})$ (for example, the smallest one), then $x \in S_1$ iff $\phi(x) \in T$. The set K can also be shown to be 1-1 complete, but the proof is a bit trickier.

If $S_1 \leq_m S_2$ and S_2 is r.e. (recursive), then S_1 is also r.e. (recursive). An r.e. m-complete set cannot be recursive: if S were an m-complete recursive set, then $K \leq_m S$, and this would imply that K is recursive, which it is not.

Some questions about reducibility naturally arise. First, is there an r.e. set which is neither recursive nor many-one complete? Emil Post answered this question in the affirmative, and we shall prove his result later on in the course. Second, are there any many-one complete sets that are not 1-1 complete? The answer is no; this result is surprising, and the proof is nontrivial; we shall give the proof later on. (The notion of being 1-complete and of being m-complete are also equivalent to the notion of being *creative*, which we shall define later on.)

Despite Post's result, all (or practically all) naturally arising r.e. sets are either 1-1 complete or recursive. That is, while r.e. sets that are neither 1-1 complete nor recursive exist in great abundance, they tend to arise as cooked-up counterexamples rather than sets which are interesting for separate reasons. A common way to prove that an r.e. set is nonrecursive is to show that some 1-1 complete set reduces to it, which implies that it is 1-1 complete.

Another way to put this is in terms of degrees. Among the r.e. 1-1 degrees (i.e. $\equiv_1$-equivalence classes containing r.e. sets), there is a degree on top (the degree of 1-1 complete sets), and, excluding the degrees containing finite and cofinite sets, a degree on the bottom (the degree of recursive sets with infinite complements), and many degrees in between. However, all the naturally occurring r.e. sets are to be found on top or on the bottom.

Besides $\leq_m$ and $\leq_1$, there are coarser-grained reducibility relations, all of which give an intuitive notion of the idea that given an oracle for a set B, we can decide A. Post, who originally formulated the notions of many-one and 1-1 reducibilities, gave a variety of reducibility notions, still studied today. One of his notions, the broadest of all, was supposed to capture the intuitive notion of being able to decide A given an oracle that will answer all questions about set membership in B. He called this notion 'Turing-reducibility'; it has also been called 'relative recursiveness' and 'recursiveness in'. As we said above, Post found an r.e. set that was not many-one complete (and therefore not 1-1 complete). However, he was able to define another reducibility relation with respect to which this set was still complete. In general, he found broader and broader reducibility relations, and more complicated r.e. but not recursive sets that failed to be complete with respect to them. However, he could not solve this problem for the basic notion of Turing-reducibility, and it was a long-standing question whether there are any r.e. sets which are neither recursive nor Turing-complete. This was answered in the affirmative in 1956 by Friedberg and Mucnik.

Two sets A and B are called *recursively isomorphic* (in symbols, $A \equiv B$) if there is a 1-1 total recursive function ϕ which maps **N** *onto* **N**, and such that $B = \{\phi(x): x \in A\}$. (It follows that ϕ^{-1} is also such a function and that $A = \{\phi^{-1}(x): x \in B\}$.) If $A \equiv B$, then it is easy to see that $A \equiv_1 B$, since ϕ: $A \leq_1 B$ and ϕ^{-1}: $B \leq_1 A$. The converse is also true, and is highly nontrivial. It was once proposed that recursion theory be regarded as the study of those properties of sets of natural numbers which are invariant under recursive isomorphism. In fact, nearly all the properties studied by recursion theory are of this nature; however, there are some exceptions.

The Relation of Substitution

In several occasions we have mentioned the relation of substitution of a term for all the free occurrences of a variable in a formula, noting that we could use it to give alternative proofs of some results. For example, we could have given an alternative proof that RE defines its own satisfaction using definitions of the notion of formula and of the relation of substitution. Also, we could have defined truth in RE by means of satisfaction in RE using substitution, instead of Tarski's trick. We will now show how a certain notion of "naive substitution" is definable within RE, leaving as an exercise showing how to define the notion of proper substitution.

First, we will define a relation that we will call "term substitution": specifically, we shall define the relation $\{<t_1, t_2, v, t>$: the term t_2 comes from the term t_1 by replacing all occurrences of the variable v, if any, by the term t; if v does not occur in t_1, then $t_1=t_2\}$. This is defined by the following formula of RE:

$$\begin{gathered}\mathrm{TSubst}(t_1, t_2, v, t) =_{\mathrm{df.}} \mathrm{Vbl}(v) \wedge \mathrm{Term}(t_1) \wedge \mathrm{Term}(t_2) \wedge \mathrm{Term}(t) \wedge [((j\leq t_1)\sim([j,v]\in t_1) \wedge t_1=t_2)\\ \vee ((\exists j\leq t_1)([j+1,v]\in t_1 \wedge (\exists l\leq t)(\mathrm{Seql}(t,l+1) \wedge \mathrm{Seql}(t_2,j+l+1) \wedge (y\leq t)([l+1,y]\in t \supset [j+l,y]\in t_2)\\ \wedge (i\leq j+l)([i, [\mathbf{0}^{(4)}, [\mathbf{0}^{(1)},\mathbf{0}^{(1)}]]]\in t_2))))].\end{gathered}$$

(Recall that all terms of RE are either of the form $f^1_1...f^1_1\mathbf{0}$ or of the form $f^1_1...f^1_1x_i$, for some i.)

We now show how to define the notion of "naive substitution", that is the relation $\{<m_1, m_2, v, m>$: the sequence m_2 comes from the sequence m_1 by replacing all occurrences of the variable v by the term m$\}$. We call this "naive" substitution because the result of a substitution of this kind may not be a formula, even if the expression operated upon was one (note, for example, that even occurrences of the variable to be replaced within quantifiers will be replaced, so if the replacing term is not a variable, substitution may transform a quantifier into an expression that cannot form part of a formula.) Naive substitution is definable by the following formula of RE:

$$\begin{gathered}\mathrm{NSubst}(m_1,m_2,v,m)=_{\mathrm{df.}} (\exists l\leq m_1)(\mathrm{Seql}(m_1,l) \wedge \mathrm{Seql}(m_2,l) \wedge (i\leq l)(y\leq l)\big[(([i,y]\in m_1 \wedge\\ \sim(\exists j\leq y)([j,v]\in y)) \supset [i,y]\in m_2) \wedge (([i,y]\in m_1 \wedge (\exists j\leq y)([j,v]\in y)) \supset (z\leq m_2)(\mathrm{TSubst}(y,z,v,m)\\ \supset [i,z]\in m_2))\big]).\end{gathered}$$

This simply "says" that if y is a part of m_1 in which v does not occur, it's left untouched in m_2, and if it is a term in which v occurs, v is replaced by m in y to obtain a term z which is then part of the result m_2.

The result of a naive substitution will not in general be a formula. It will be a formula, however, if the variable to be replaced never occurs bound in the initial formula (and also if the term replacing the variable in a formula is another variable). The utility of naive substitution can be seen from the fact that it is sufficient for showing that the set of logical axioms (of a given language) in the deductive system of the next section is r.e. Notice, in particular, that axiom schema 4 only invokes naive substitution. In standard systems, in place of axiom schemata 4 and 5 we find schemata 4' and 5', and the natural way of coding these is more complicated. It involves proper substitution, that is, the relation $\{<m_1, m_2, v, m>$: the sequence m_2 comes from the sequence m_1 by replacing all *free* occurrences of the variable v by the term t; and no variable occurring in m becomes bound in $m_2\}$. The definition of proper substitution in RE is left as an exercise.

Deductive Systems.

We want, for a given language L, a deductive system in which all and only the valid sentences of L are provable. When L does not contain function symbols, the following is such a system.

The axioms are all of the instances (in L) of the following schemata:

1. $A \supset (B \supset A)$;

2. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$;

3. $(\sim A \supset \sim B) \supset (B \supset A)$;

4. $(x_i)A \supset A'$, where A' is got from A by substituting all occurrences of x_i in A by a fixed term t, and neither (x_i) nor (x_j) occurs in A, where x_j is any variable occurring in t;

5. $A \supset (x_i)A$, where x_i does not occur in A;

6. $(x_i)(A \supset B) \supset ((x_i)A \supset (x_i)B)$.

There are also two inference rules:

modus ponens (MP): $A \supset B, A$ universal generalization (UG): A

$$\frac{}{B} \qquad\qquad\qquad \frac{}{(x_i)A}$$

A *proof* in L is a finite sequence of formulae of L such that each formula is either an axiom or follows from earlier formulae by one of the inference rules. A sentence is a theorem just in case it occurs in a proof. More generally, a proof from Γ, where Γ is a set of *sentences* of L, is a finite sequence of formulae in which each formula is either an axiom *or an element of Γ*, or follows from earlier formulae by the inference rules. A sentence is a theorem of Γ just in case it occurs in a proof from Γ. We write fi A to mean that A is a theorem, and Γ fi A to mean that A is a theorem of Γ.

Remarks. The axiom schemata 1-3, along with MP, are sufficient to prove all tautologies. We could have simply taken all tautologies (of L) as axioms, but the present approach will prove more convenient later.

The present system differs from standard systems in that axiom schemata 4 and 5 are usually formulated as follows:

4'. $(x_i)A \supset A'$, where A' is got from A by replacing all *free* occurrences of x_i in A by a fixed term t, where either t is a constant, or t is a variable x_j and no free occurrence of x_i in A falls within the scope of a quantifier (x_j).

5'. $A \supset (x_i)A$, where x_i does not occur *free* in A.

All instances of 4' and 5' that are not also instances of 4 and 5 prove to be derivable and hence redundant. As we said, 4 and 5 are simpler to represent in our coding of syntax than 4' and 5'.

(In any deductive system which contains a version of universal instantiation, some restriction like that in 4 or 4' must be made, for the *prima facie* more natural scheme

$(x_i)A \supset A'$, where A' comes from A by replacing all free occurrences of x_i in A by a fixed term t

is invalid. To see this, consider the instance $(x)(\exists y)\ x \neq y \supset (\exists y)\ y \neq y$. In any interpretation whose domain has more than one element and in which = is interpreted as identity, this sentence is false. The problem is that the instantial term, y, becomes bound once it is substituted for x; as long as we prevent this sort of thing, the restricted scheme will be valid.)

We say that a system is *sound* just in case every theorem is valid, and every theorem of Γ is a consequence of Γ, for any Γ. A system is *complete* if every valid sentence is a theorem, and *strongly complete* if for all Γ, every consequence of Γ is a theorem of Γ. The proof that our system is sound is fairly easy by induction on the length of a proof of any given theorem. We will sketch later the argument for completeness (and strong completeness), with more discussion of variant formulations, and in particular of the comprehensive virtues and drawbacks of 4 and 5 as opposed to 4' and 5'. In particular, how to derive 4' and 5' from 4 and 5 will be sketched later.

The Narrow and Broad Languages of Arithmetic.

By the *narrow* language of arithmetic, we mean the language as given in lecture I. Recall that the primitive connectives of that language are $\supset$ and ~ and that (x_i) is the only primitive quantifier; so, perversely, the logical vocabulary of the narrow language is disjoint from that of RE. By the *broad* language of arithmetic we mean the language which has all the vocabulary of the narrow language, and in addition the primitive connectives $\wedge$ and $\vee$ and the quantifiers $(\exists x_i)$, $(x_i < t)$, and $(\exists x_i < t)$ (with the usual restrictions on t). So the broad language of arithmetic literally contains the languages RE and Lim (but not Lim^+). We shall use L ambiguously to refer to both the broad and the narrow language of arithmetic; when we wish to refer to them unambiguously, we shall use L^- to refer to the narrow language and L^+ to refer to the broad language. Note that L^- and L^+ are equal in expressive power, since the extra connectives of L^+ are already definable in L^-.

The language L^+ has redundancies, as its extra connectives and quantifiers are already definable in L^-. L^- also has redundancies. For example, the negation sign is superfluous in L^-, as ~A is equivalent to $A \supset \mathbf{0} = \mathbf{0}'$. If we had included the function symbols + and ·, then all the connectives would have been superfluous, since they could be eliminated in the manner indicated in an exercise. Even in L^+, we can eliminate all of the connectives except for $\wedge$. To see this, let A be any formula of L^+, and let A* be an equivalent formula in which A and M are replaced by + and ·. Let A** be an equivalent formula in which no connectives appear, constructed in the way indicated in lecture II. Finally, let A*** be a formula of L+ got from A** by Russell's trick. Since the only connective that Russell's trick introduces is $\wedge$, $\wedge$ is the only connective A*** contains; and A*** is equivalent to A.

Note that our deductive system only has axioms and rules for the connectives ~ and $\supset$ and the quantifier (x_i); when we are considering the broad language of arithmetic, we want our system to prove all the valid formulae that contain the new logical vocabulary as well as the old. This can be achieved by adding the following equivalence axiom schemes when we are working in L^+:

$(A \vee B) \equiv (\sim A \supset B)$
$(A \wedge B) \equiv \sim(A \supset \sim B)$
$(\exists x_i)A \equiv \sim(x_i)\sim A$
$(x_i < t)A \equiv (x_i)(\text{Less}(x_i, t) \supset A)$

$(\exists x_i < t)A \equiv (\exists x_i)(\mathrm{Less}(x_i, t) \wedge A)$

(Note that $\equiv$ is a defined rather than a primitive symbol, even in L^+.) It can be shown (though we will not show it here) that when these schemes are taken as axioms, every formula of L^+ is provably equivalent to a formula of L^-, i.e. for every formula A of L^+ there is a formula A' of L^- such that $A \equiv A'$ is provable. Thus, the completeness of the system of L^+ follows from that for L^-.

It is easy to show that the set of logical axioms for L (either L^+ or L^-) is r.e. We can then prove, using the Generated Sets Theorem, that the set of provable formulae is r.e., and that if Γ is an r.e. set of sentences of L, then the set of theorems of Γ is r.e. We can generalize this to languages other than L. If K is a first order language such that the set of formulae of K is r.e., then the set of provable formulae in K is r.e., and if Γ is an r.e. set of sentences of K, then the set of theorems of Γ is r.e. For the set of formulae of K to be r.e., it is necessary and sufficient that the set $\{i: a_i \in K\}$ and the relations $\{<n, i>: P^n_i \in K\}$ and $\{<n, i>: f^n_i \in K\}$ be r.e.

The Theories Q and PA.

There are two theories in the language L that are traditionally given special attention. One is the theory Q, also called *Robinson's Arithmetic* (after its inventor Raphael Robinson). Q is usually given in the language with + and ·, so our version of it is slightly nonstandard. In the usual version, the axioms of Q are

1. $(x_1)\ \mathbf{0} \neq x_1'$
2. $(x_1)(x_2)\ (x_1' = x_2' \supset x_1 = x_2)$
3. $(x_1)\ (x_1 = \mathbf{0} \vee (\exists x_2)\ x_1 = x_2')$
4. $(x_1)\ x_1 + \mathbf{0} = x_1$
5. $(x_1)(x_2)\ x_1 + x_2' = (x_1 + x_2)'$
6. $(x_1)\ x_1 \cdot \mathbf{0} = \mathbf{0}$
7. $(x_1)(x_2)\ x_1 \cdot (x_2') = x_1 \cdot x_2 + x_1$

(Axioms 4-7 are usually called the *recursion axioms*.) To adapt this to our language, we rewrite the axioms as follows:

1. $(x_1)\ \mathbf{0} \neq x_1'$
2. $(x_1)(x_2)\ (x_1' = x_2' \supset x_1 = x_2)$
3. $(x_1)\ (x_1 = \mathbf{0} \vee (\exists x_2)\ x_1 = x_2')$
4. $(x_1)\ A(x_1, \mathbf{0}, x_1)$
5. $(x_1)(x_2)(x_3)\ (A(x_1, x_2, x_3) \supset A(x_1, x_2', x_3'))$

6. $(x_1)\ M(x_1, \mathbf{0}, \mathbf{0})$
7. $(x_1)(x_2)(x_3)(x_4)\ ((M(x_1, x_2, x_3) \wedge A(x_3, x_1, x_4) \supset M(x_1, x_2', x_4))$

(Note that axioms 1-3 are unchanged.) In addition, we need existence and uniqueness axioms for addition and multiplication:

$(x_1)(x_2)(\exists x_3)\ (A(x_1, x_2, x_3) \wedge (x_4)(A(x_1, x_2, x_4) \supset x_4 = x_3))$
$(x_1)(x_2)(\exists x_3)\ (M(x_1, x_2, x_3) \wedge (x_4)(M(x_1, x_2, x_4) \supset x_4 = x_3))$

(These are unnecessary for Q as it is usually stated, because their analogs in the language with function symbols + and · rather than predicates A and M are logical truths.) Finally, we did not include axioms for identity in our deductive system for the predicate calculus, so we must include them here. The usual identity axioms are the reflexivity axiom $(x_1)\ x_1=x_1$ and the axiom scheme $(x_1)(x_2)\ (x_1 = x_2 \supset A \equiv A')$, where A is any formula with at most x_1 and x_2 free and A' comes from A by replacing one or more free occurrences of x_1 by x_2. In fact, we can get away with taking only finitely many instances of this scheme as axioms, and the rest will be deducible. Specifically, we can take as our identity axioms the reflexivity axiom and those instances of the above scheme in which A is an atomic formula not containing the function symbol '. Since there are only finitely many predicates in L, there are only finitely many such instances. Q, then, is the theory in L whose axioms are 1-7 above along with the existence and uniqueness clauses and the identity axioms just specified.

PA, or *Peano Arithmetic*, comes from Q by deleting axiom 3 and adding all those sentences which are instances in L of the *induction scheme*:

$$[A(\mathbf{0}) \wedge (x_1)(A(x_1) \supset A(x_1'))] \supset (x_1)A(x_1).$$

(Axiom 3 is a theorem of the resulting system, so we need not take it as an axiom.)

The intuitive idea behind the induction scheme is that if zero has a property, and if whenever a number n has that property n' does too, then every number has that property. This was the intuition that Peano, and Dedekind before him, intended to capture, through an induction *axiom*, that we could formalize

$$(P)([P(\mathbf{0}) \wedge (x_1)(P(x_1) \supset P(x_1'))] \supset (x_1)P(x_1)).$$

However, since in our languages we do not have quantification over arbitrary sets of natural numbers, the induction axiom cannot be formalized in them. Unlike Dedekind's and Peano's axiom, the induction scheme of the system we call 'Peano Arithmetic' only guarantees that when zero has a property definable in L and when a number n has it n' does too, then every number has it. So the induction scheme is really weaker than the intuitive idea behind it, that

Dedekind and Peano had in mind. In this respect, the name 'Peano Arithmetic' is somewhat misleading.

The theory PA is adequate for elementary number theory, in the sense that all of elementary number theory can be carried out within PA. This is not obvious, however, and requires proof. Notice, for example, that before the work of Gödel, it was not obvious that such simple functions as exponentiation could even be defined in the language in which PA is given, and exponentiation should certainly be regarded as a part of elementary number theory. This illustrates another respect in which the name 'Peano Arithmetic' is misleading, since it suggests that elementary arithmetic can be developed in that system in a straightforward, obvious manner.

Exercises

1. Consider a language RE^{exp} which has the same terms, connectives and quantifiers as RE but has only one predicate letter P_1^3. P_1^3xyz is interpreted as $x^y=z$ (it doesn't matter what to say about the case 0^0; you can call P_1^3xyz always false in that case, or give it the value 0 or 1). Prove that RE^{exp} defines the same sets and relations as RE. Prove also that in RE^{exp} (for the same reason as in RE) disjunction is superfluous. (Remark: half of this exercise has been done. To do the other direction, that is, defining the notions of RE in RE^{exp}, it is best to proceed in the opposite order from what appears to be natural.)

2. Cantor proved that there can be no function ϕ mapping a set onto the set of all of its subsets. Show directly that if there were such a mapping, then we would have an interpretation of '$\in$' which makes true the unrestricted comprehension schema, including the version with parameters. Remark: hence, any set-theoretical paradox that proves the inconsistency of the schema also proves the theorem of Cantor in question. The case Cantor actually used was once again the analog of Russell's paradox. Historically, this went the other way around, since Russell discovered his paradox by analyzing Cantor's proof.

3. Show that the relations $\leq_m$ and $\leq_1$ are reflexive and transitive.

4. Show that if $A \leq_m B$ and B is r.e. (recursive), then A is r.e. (recursive).

5. For the language of arithmetic, prove using the Generated Sets Theorem that if a set of axioms is r.e., then the set of theorems logically provable from it is r.e.