Chapter udf

Arithmetization of Syntax

Note that arithmetization for signed tableaux is not yet available.

art.1 Introduction

inc:art:int:sec In order to connect computability and logic, we need a way to talk about the objects of logic (symbols, terms, formulas, derivations), operations on them, and their properties and relations, in a way amenable to computational treatment. We can do this directly, by considering computable functions and relations on symbols, sequences of symbols, and other objects built from them. Since the objects of logical syntax are all finite and built from an enumerable sets of symbols, this is possible for some models of computation. But other models of computation—such as the recursive functions—-are restricted to numbers, their relations and functions. Moreover, ultimately we also want to be able to deal with syntax within certain theories, specifically, in theories formulated in the language of arithmetic. In these cases it is necessary to *arithmetize* syntax, i.e., to represent syntactic objects, operations on them, and their relations, as numbers, arithmetical functions, and arithmetical relations, respectively. The idea, which goes back to Leibniz, is to assign numbers to syntactic objects.

It is relatively straightforward to assign numbers to symbols as their "codes." Some symbols pose a bit of a challenge, since, e.g., there are infinitely many variables, and even infinitely many function symbols of each arity n. But of course it's possible to assign numbers to symbols systematically in such a way that, say, v_2 and v_3 are assigned different codes. Sequences of symbols (such as terms and formulas) are a bigger challenge. But if we can deal with sequences of numbers purely arithmetically (e.g., by the powers-of-primes coding of sequences), we can extend the coding of individual symbols to coding of sequences of symbols, and then further to sequences or other arrangements of formulas, such as derivations. This extended coding is called "Gödel numbering." Every term, formula, and derivation is assigned a Gödel number.

By coding sequences of symbols as sequences of their codes, and by chosing a system of coding sequences that can be dealt with using computable functions, we can then also deal with Gödel numbers using computable functions. In practice, all the relevant functions will be primitive recursive. For instance, computing the length of a sequence and computing the i-th element of a sequence from the code of the sequence are both primitive recursive. If the number coding the sequence is, e.g., the Gödel number of a formula φ, we immediately see that the length of a formula and the (code of the) i-th symbol in a formula can also be computed from the Gödel number of φ. It is a bit harder to prove that, e.g., the property of being the Gödel number of a correctly formed term or of a correct derivation is primitive recursive. It is nevertheless possible, because the sequences of interest (terms, formulas, derivations) are inductively defined.

As an example, consider the operation of substitution. If φ is a formula, x a variable, and t a term, then $\varphi[t/x]$ is the result of replacing every free occurrence of x in φ by t. Now suppose we have assigned Gödel numbers to φ, x, t—say, k, l, and m, respectively. The same scheme assigns a Gödel number to $\varphi[t/x]$, say, n. This mapping—of k, l, and m to n—is the arithmetical analog of the substitution operation. When the substitution operation maps φ, x, t to $\varphi[t/x]$, the arithmetized substitution functions maps the Gödel numbers k, l, m to the Gödel number n. We will see that this function is primitive recursive.

Arithmetization of syntax is not just of abstract interest, although it was originally a non-trivial insight that languages like the language of arithmetic, which do not come with mechanisms for "talking about" languages can, after all, formalize complex properties of expressions. It is then just a small step to ask what a theory in this language, such as Peano arithmetic, can *prove* about its own language (including, e.g., whether sentences are provable or true). This leads us to the famous limitative theorems of Gödel (about unprovability) and Tarski (the undefinability of truth). But the trick of arithmetizing syntax is also important in order to prove some important results in computability theory, e.g., about the computational power of theories or the relationship between different models of computability. The arithmetization of syntax serves as a model for arithmetizing other objects and properties. For instance, it is similarly possible to arithmetize configurations and computations (say, of Turing machines). This makes it possible to simulate computations in one model (e.g., Turing machines) in another (e.g., recursive functions).

art.2 Coding Symbols

The basic language $\mathcal{L}$ of first order logic makes use of the symbols inc:art:cod:sec

$$\bot \quad \neg \quad \vee \quad \wedge \quad \rightarrow \quad \forall \quad \exists \quad = \quad (\quad) \quad ,$$

together with enumerable sets of variables and constant symbols, and enumerable sets of function symbols and predicate symbols of arbitrary arity. We can assign *codes* to each of these symbols in such a way that every symbol is as-

signed a unique number as its code, and no two different symbols are assigned the same number. We know that this is possible since the set of all symbols is enumerable and so there is a bijection between it and the set of natural numbers. But we want to make sure that we can recover the symbol (as well as some information about it, e.g., the arity of a function symbol) from its code in a computable way. There are many possible ways of doing this, of course. Here is one such way, which uses primitive recursive functions. (Recall that $\langle n_0, \dots, n_k \rangle$ is the number coding the sequence of numbers $n_0, \dots, n_k$.)

Definition art.1. If s is a symbol of $\mathcal{L}$, let the *symbol code* c_s be defined as follows:

1. If s is among the logical symbols, c_s is given by the following table:

$\bot$	$\neg$	$\vee$	$\wedge$	$\rightarrow$	$\forall$
$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$	$\langle 0, 4 \rangle$	$\langle 0, 5 \rangle$
$\exists$	$=$	$($	$)$	$,$	
$\langle 0, 6 \rangle$	$\langle 0, 7 \rangle$	$\langle 0, 8 \rangle$	$\langle 0, 9 \rangle$	$\langle 0, 10 \rangle$	

2. If s is the i-th variable v_i, then $\mathrm{c}_s = \langle 1, i \rangle$.

3. If s is the i-th constant symbol c_i, then $\mathrm{c}_s = \langle 2, i \rangle$.

4. If s is the i-th n-ary function symbol f_i^n, then $\mathrm{c}_s = \langle 3, n, i \rangle$.

5. If s is the i-th n-ary predicate symbol P_i^n, then $\mathrm{c}_s = \langle 4, n, i \rangle$.

Proposition art.2. *The following relations are primitive recursive:*

1. $\mathrm{Fn}(x, n)$ *iff* x *is the code of* f_i^n *for some* i*, i.e.,* x *is the code of an* n*-ary function symbol.*

2. $\mathrm{Pred}(x, n)$ *iff* x *is the code of* P_i^n *for some* i *or* x *is the code of* $=$ *and* $n = 2$*, i.e.,* x *is the code of an* n*-ary predicate symbol.*

Definition art.3. If $s_0, \dots, s_{n-1}$ is a sequence of symbols, its *Gödel number* is $\langle \mathrm{c}_{s_0}, \dots, \mathrm{c}_{s_{n-1}} \rangle$.

Note that *codes* and *Gödel numbers* are different things. For instance, the variable v_5 has a code $\mathrm{c}_{v_5} = \langle 1, 5 \rangle = 2^2 \cdot 3^6$. But the variable v_5 considered as a term is also a sequence of symbols (of length 1). The *Gödel number* ${}^{\#}v_5{}^{\#}$ of the *term* v_5 is $\langle \mathrm{c}_{v_5} \rangle = 2^{\mathrm{c}_{v_5}+1} = 2^{2^2 \cdot 3^6 + 1}$. explanation

Example art.4. Recall that if $k_0, \dots, k_{n-1}$ is a sequence of numbers, then the code of the sequence $\langle k_0, \dots, k_{n-1} \rangle$ in the power-of-primes coding is

$$2^{k_0+1} \cdot 3^{k_1+1} \cdot \dots \cdot p_{n-1}^{k_{n-1}},$$

where p_i is the i-th prime (starting with $p_0 = 2$). So for instance, the formula $v_0 = \mathsf{o}$, or, more explicitly, $=(v_0, c_0)$, has the Gödel number

$$\langle \mathrm{c}_=, \mathrm{c}_(, \mathrm{c}_{v_0}, \mathrm{c}_,, \mathrm{c}_{c_0}, \mathrm{c}_) \rangle.$$

Here, $\mathrm{c}_=$ is $\langle 0, 7 \rangle = 2^{0+1} \cdot 3^{7+1}$, c_{v_0} is $\langle 1, 0 \rangle = 2^{1+1} \cdot 3^{0+1}$, etc. So $\#{=}(v_0, c_0)\#$ is

$$\begin{aligned} &2^{\mathrm{c}_=+1} \cdot 3^{\mathrm{c}_(+1} \cdot 5^{\mathrm{c}_{v_0}+1} \cdot 7^{\mathrm{c}_,+1} \cdot 11^{\mathrm{c}_{c_0}+1} \cdot 13^{\mathrm{c}_)+1} = \\ &\qquad 2^{2^1 \cdot 3^8+1} \cdot 3^{2^1 \cdot 3^9+1} \cdot 5^{2^2 \cdot 3^1+1} \cdot 7^{2^1 \cdot 3^{11}+1} \cdot 11^{2^3 \cdot 3^1+1} \cdot 13^{2^1 \cdot 3^{10}+1} = \\ &\qquad\qquad 2^{13\,123} \cdot 3^{39\,367} \cdot 5^{13} \cdot 7^{354\,295} \cdot 11^{25} \cdot 13^{118\,099}. \end{aligned}$$

art.3 Coding Terms

explanation A term is simply a certain kind of sequence of symbols: it is built up inductively from constants and variables according to the formation rules for terms. Since sequences of symbols can be coded as numbers—using a coding scheme for the symbols plus a way to code sequences of numbers—assigning Gödel numbers to terms is not difficult. The challenge is rather to show that the property a number has if it is the Gödel number of a correctly formed term is computable, or in fact primitive recursive. inc:art:trm:sec

Variables and constant symbols are the simplest terms, and testing whether x is the Gödel number of such a term is easy: $\mathrm{Var}(x)$ holds if x is $\#v_i\#$ for some i. In other words, x is a sequence of length 1 and its single element $(x)_0$ is the code of some variable v_i, i.e., x is $\langle\langle 1, i \rangle\rangle$ for some i. Similarly, $\mathrm{Const}(x)$ holds if x is $\#c_i\#$ for some i. Both of these relations are primitive recursive, since if such an i exists, it must be $< x$:

$$\begin{aligned} \mathrm{Var}(x) &\Leftrightarrow (\exists i < x)\ x = \langle\langle 1, i \rangle\rangle \\ \mathrm{Const}(x) &\Leftrightarrow (\exists i < x)\ x = \langle\langle 2, i \rangle\rangle \end{aligned}$$

Proposition art.5. *The relations* $\mathrm{Term}(x)$ *and* $\mathrm{ClTerm}(x)$ *which hold iff* x *is the Gödel number of a term or a closed term, respectively, are primitive recursive.* *inc:art:trm:prop:term-primrec*

Proof. A sequence of symbols s is a term iff there is a sequence $s_0, \ldots, s_{k-1} = s$ of terms which records how the term s was formed from constant symbols and variables according to the formation rules for terms. To express that such a putative formation sequence follows the formation rules it has to be the case that, for each $i < k$, either

1. s_i is a variable v_j, or
2. s_i is a constant symbol c_j, or
3. s_i is built from n terms $t_1, \ldots, t_n$ occurring prior to place i using an n-place function symbol f_j^n.

To show that the corresponding relation on Gödel numbers is primitive recursive, we have to express this condition primitive recursively, i.e., using primitive recursive functions, relations, and bounded quantification.

Suppose y is the number that codes the sequence $s_0, \ldots, s_{k-1}$, i.e., $y = \langle {}^{\#}s_0{}^{\#}, \ldots, {}^{\#}s_{k-1}{}^{\#}\rangle$. It codes a formation sequence for the term with Gödel number x iff for all $i < k$:

1. $\mathrm{Var}((y)_i)$, or
2. $\mathrm{Const}((y)_i)$, or
3. there is an n and a number $z = \langle z_1, \ldots, z_n\rangle$ such that each z_l is equal to some $(y)_{i'}$ for $i' < i$ and

$$(y)_i = {}^{\#}f^n_j({}^{\#} \frown \mathrm{flatten}(z) \frown {}^{\#})^{\#},$$

and moreover $(y)_{k-1} = x$. (The function $\mathrm{flatten}(z)$ turns the sequence $\langle {}^{\#}t_1{}^{\#}, \ldots, {}^{\#}t_n{}^{\#}\rangle$ into ${}^{\#}t_1, \ldots, t_n{}^{\#}$ and is primitive recursive.)

The indices j, n, the Gödel numbers z_l of the terms t_l, and the code z of the sequence $\langle z_1, \ldots, z_n\rangle$, in (3) are all less than y. We can replace k above with $\mathrm{len}(y)$. Hence we can express "y is the code of a formation sequence of the term with Gödel number x" in a way that shows that this relation is primitive recursive.

We now just have to convince ourselves that there is a primitive recursive bound on y. But if x is the Gödel number of a term, it must have a formation sequence with at most $\mathrm{len}(x)$ terms (since every term in the formation sequence of s must start at some place in s, and no two subterms can start at the same place). The Gödel number of each subterm of s is of course $\leq x$. Hence, there always is a formation sequence with code $\leq p_{k-1}^{k(x+1)}$, where $k = \mathrm{len}(x)$.

For ClTerm, simply leave out the clause for variables. □

Problem art.1. Show that the function $\mathrm{flatten}(z)$, which turns the sequence $\langle {}^{\#}t_1{}^{\#}, \ldots, {}^{\#}t_n{}^{\#}\rangle$ into ${}^{\#}t_1, \ldots, t_n{}^{\#}$, is primitive recursive.

inc:art:trm:prop:num-primrec **Proposition art.6.** *The function* $\mathrm{num}(n) = {}^{\#}\overline{n}^{\#}$ *is primitive recursive.*

Proof. We define $\mathrm{num}(n)$ by primitive recursion:

$$\begin{aligned} \mathrm{num}(0) &= {}^{\#}\mathsf{o}^{\#} \\ \mathrm{num}(n+1) &= {}^{\#}\mathsf{\prime}({}^{\#} \frown \mathrm{num}(n) \frown {}^{\#})^{\#}. \end{aligned}$$

□

art.4 Coding Formulas

inc:art:frm:sec **Proposition art.7.** *The relation* $\mathrm{Atom}(x)$ *which holds iff* x *is the Gödel number of an atomic formula, is primitive recursive.*

Proof. The number x is the Gödel number of an atomic formula iff one of the following holds:

1. There are n, $j < x$, and $z < x$ such that for each $i < n$, $\mathrm{Term}((z)_i)$ and $x =$
$$\#\mathsf{P}^n_j(\# \frown \mathrm{flatten}(z) \frown \#)\#.$$
2. There are $z_1, z_2 < x$ such that $\mathrm{Term}(z_1)$, $\mathrm{Term}(z_2)$, and $x =$
$$\#{=}(\# \frown z_1 \frown \#,\# \frown z_2 \frown \#)\#.$$
3. $x = \#\bot\#$.
4. $x = \#\top\#$. □

Proposition art.8. *The relation* $\mathrm{Frm}(x)$ *which holds iff* x *is the Gödel number of a formula is primitive recursive.* inc:art:frm: prop:frm-primrec

Proof. A sequence of symbols s is a formula iff there is formation sequence s_0, $\ldots$, $s_{k-1} = s$ of formula which records how s was formed from atomic formulas according to the formation rules. The code for each s_i (and indeed of the code of the sequence $\langle s_0, \ldots, s_{k-1}\rangle$) is less than the code x of s. □

Problem art.2. Give a detailed proof of Proposition art.8 along the lines of the first proof of Proposition art.5.

Proposition art.9. *The relation* $\mathrm{FreeOcc}(x, z, i)$*, which holds iff the* i*-th symbol of the formula with Gödel number* x *is a free occurrence of the variable with Gödel number* z*, is primitive recursive.* inc:art:frm: prop:freeocc-primrec

Proof. Exercise. □

Problem art.3. Prove Proposition art.9. You may make use of the fact that any substring of a formula which is a formula is a sub-formula of it.

Proposition art.10. *The property* $\mathrm{Sent}(x)$ *which holds iff* x *is the Gödel number of a sentence is primitive recursive.*

Proof. A sentence is a formula without free occurrences of variables. So $\mathrm{Sent}(x)$ holds iff

$$(\forall i < \mathrm{len}(x))\ (\forall z < x)\ ((\exists j < z)\ z = \#\mathsf{v}_j\# \rightarrow \neg\mathrm{FreeOcc}(x, z, i)).$$ □

art.5 Substitution

inc:art:sub:sec Recall that substitution is the operation of replacing all free occurrences of a variable u in a formula φ by a term t, written $\varphi[t/u]$. This operation, when carried out on Gödel numbers of variables, formulas, and terms, is primitive recursive.

inc:art:sub:prop:subst-primrec **Proposition art.11.** *There is a primitive recursive function* $\mathrm{Subst}(x, y, z)$ *with the property that*

$$\mathrm{Subst}({}^{\#}\varphi^{\#}, {}^{\#}t^{\#}, {}^{\#}u^{\#}) = {}^{\#}\varphi[t/u]^{\#}.$$

Proof. We can then define a function hSubst by primitive recursion as follows:

$$\begin{aligned} \mathrm{hSubst}(x, y, z, 0) &= \Lambda \\ \mathrm{hSubst}(x, y, z, i+1) &= \\ &\begin{cases} \mathrm{hSubst}(x, y, z, i) \frown y & \text{if } \mathrm{FreeOcc}(x, z, i) \\ \mathrm{append}(\mathrm{hSubst}(x, y, z, i), (x)_i) & \text{otherwise.} \end{cases} \end{aligned}$$

$\mathrm{Subst}(x, y, z)$ can now be defined as $\mathrm{hSubst}(x, y, z, \mathrm{len}(x))$. □

inc:art:sub:prop:free-for **Proposition art.12.** *The relation* $\mathrm{FreeFor}(x, y, z)$*, which holds iff the term with Gödel number* y *is free for the variable with Gödel number* z *in the formula with Gödel number* x*, is primitive recursive.*

Proof. Exercise. □

Problem art.4. Prove Proposition art.12

art.6 Derivations in LK

inc:art:plk:sec In order to arithmetize derivations, we must represent derivations as numbers. explanation Since derivations are trees of sequents where each inference carries also a label, a recursive representation is the most obvious approach: we represent a derivation as a tuple, the components of which are the end-sequent, the label, and the representations of the sub-derivations leading to the premises of the last inference.

Definition art.13. If Γ is a finite sequence of sentences, $\Gamma = \langle \varphi_1, \ldots, \varphi_n \rangle$, then ${}^{\#}\Gamma^{\#} = \langle {}^{\#}\varphi_1{}^{\#}, \ldots, {}^{\#}\varphi_n{}^{\#} \rangle$.

If $\Gamma \Rightarrow \Delta$ is a sequent, then a Gödel number of $\Gamma \Rightarrow \Delta$ is

$${}^{\#}\Gamma \Rightarrow \Delta^{\#} = \langle {}^{\#}\Gamma^{\#}, {}^{\#}\Delta^{\#} \rangle$$

If π is a derivation in **LK**, then ${}^{\#}\pi^{\#}$ is defined as follows:

1. If π consists only of the initial sequent $\Gamma \Rightarrow \Delta$, then ${}^{\#}\pi^{\#}$ is

$$\langle 0, {}^{\#}\Gamma \Rightarrow \Delta^{\#} \rangle.$$

2. If π ends in an inference with one or two premises, has $\Gamma \Rightarrow \Delta$ as its conclusion, and π_1 and π_2 are the immediate subproof ending in the premise of the last inference, then ${}^{\#}\pi^{\#}$ is

$$\langle 1, {}^{\#}\pi_1{}^{\#}, {}^{\#}\Gamma \Rightarrow \Delta^{\#}, k \rangle \text{ or}$$
$$\langle 2, {}^{\#}\pi_1{}^{\#}, {}^{\#}\pi_2{}^{\#}, {}^{\#}\Gamma \Rightarrow \Delta^{\#}, k \rangle,$$

respectively, where k is given by the following table according to which rule was used in the last inference:

Rule:	WL	WR	CL	CR	XL	XR
k:	1	2	3	4	5	6
Rule:	$\neg$L	$\neg$R	$\wedge$L	$\wedge$R	$\vee$L	$\vee$R
k:	7	8	9	10	11	12
Rule:	$\to$L	$\to$R	$\forall$L	$\forall$R	$\exists$L	$\exists$R
k:	13	14	15	16	17	18
Rule:	Cut	$=$				
k:	19	20				

Example art.14. Consider the very simple derivation

$$\dfrac{\dfrac{\varphi \Rightarrow \varphi}{\varphi \wedge \psi \Rightarrow \varphi}\,\wedge\mathrm{L}}{\Rightarrow (\varphi \wedge \psi) \to \varphi}\,\to\mathrm{R}$$

The Gödel number of the initial sequent would be $p_0 = \langle 0, {}^{\#}\varphi \Rightarrow \varphi^{\#} \rangle$. The Gödel number of the derivation ending in the conclusion of $\wedge$L would be $p_1 = \langle 1, p_0, {}^{\#}\varphi \wedge \psi \Rightarrow \varphi^{\#}, 9 \rangle$ (1 since $\wedge$L has one premise, the Gödel number of the conclusion $\varphi \wedge \psi \Rightarrow \varphi$, and 9 is the number coding $\wedge$L). The Gödel number of the entire derivation then is $\langle 1, p_1, {}^{\#}\Rightarrow (\varphi \wedge \psi) \to \varphi)^{\#}, 14 \rangle$, i.e.,

$$\langle 1, \langle 1, \langle 0, {}^{\#}\varphi \Rightarrow \varphi)^{\#} \rangle, {}^{\#}\varphi \wedge \psi \Rightarrow \varphi^{\#}, 9 \rangle, {}^{\#}\Rightarrow (\varphi \wedge \psi) \to \varphi^{\#}, 14 \rangle.$$

explanation

Having settled on a representation of derivations, we must also show that we can manipulate such derivations primitive recursively, and express their essential properties and relations so. Some operations are simple: e.g., given a Gödel number p of a derivation, $\mathrm{EndSeq}(p) = (p)_{(p)_0+1}$ gives us the Gödel number of its end-sequent and $\mathrm{LastRule}(p) = (p)_{(p)_0+2}$ the code of its last rule. The property $\mathrm{Sequent}(s)$ defined by

$$\mathrm{len}(s) = 2 \wedge (\forall i < \mathrm{len}((s)_0) + \mathrm{len}((s)_1))\ \mathrm{Sent}(((s)_0 \frown (s)_1)_i)$$

holds of s iff s is the Gödel number of a sequent consisting of sentences. Some are much harder. We'll at least sketch how to do this. The goal is to show that

the relation "π is a derivation of φ from Γ" is a primitive recursive relation of the Gödel numbers of π and φ.

inc:art:plk:prop:followsby **Proposition art.15.** *The property* $\mathrm{Correct}(p)$ *which holds iff the last inference in the derivation* π *with Gödel number* p *is correct, is primitive recursive.*

Proof. $\Gamma \Rightarrow \Delta$ is an initial sequent if either there is a sentence φ such that $\Gamma \Rightarrow \Delta$ is $\varphi \Rightarrow \varphi$, or there is a term t such that $\Gamma \Rightarrow \Delta$ is $\emptyset \Rightarrow t = t$. In terms of Gödel numbers, $\mathrm{InitSeq}(s)$ holds iff

$$\begin{aligned}&(\exists x < s)\ (\mathrm{Sent}(x) \land s = \langle\langle x\rangle, \langle x\rangle\rangle) \lor \\ &(\exists t < s)\ (\mathrm{Term}(t) \land s = \langle 0, \langle {}^{\#}{=}({}^{\#} \frown t \frown {}^{\#}{,}^{\#} \frown t \frown {}^{\#})^{\#}\rangle\rangle).\end{aligned}$$

We also have to show that for each rule of inference R the relation $\mathrm{FollowsBy}_R(p)$ is primitive recursive, where $\mathrm{FollowsBy}_R(p)$ holds iff p is the Gödel number of derivation π, and the end-sequent of π follows by a correct application of R from the immediate sub-derivations of π.

A simple case is that of the $\land$R rule. If π ends in a correct $\land$R inference, it looks like this:

$$\dfrac{\begin{array}{c}\vdots\ \pi_1 \\ \Gamma \Rightarrow \Delta, \varphi\end{array} \qquad \begin{array}{c}\vdots\ \pi_2 \\ \Gamma \Rightarrow \Delta, \psi\end{array}}{\Gamma \Rightarrow \Delta, \varphi \land \psi}\ \land\mathrm{R}$$

So, the last inference in the derivation π is a correct application of $\land$R iff there are sequences of sentences Γ and Δ as well as two sentences φ and ψ such that the end-sequent of π_1 is $\Gamma \Rightarrow \Delta, \varphi$, the end-sequent of π_2 is $\Gamma \Rightarrow \Delta, \psi$, and the end-sequent of π is $\Gamma \Rightarrow \Delta, \varphi \land \psi$. We just have to translate this into Gödel numbers. If $s = {}^{\#}\Gamma \Rightarrow \Delta^{\#}$ then $(s)_0 = {}^{\#}\Gamma^{\#}$ and $(s)_1 = {}^{\#}\Delta^{\#}$. So, $\mathrm{FollowsBy}_{\land\mathrm{R}}(p)$ holds iff

$$\begin{aligned}&(\exists g < p)\ (\exists d < p)\ (\exists a < p)\ (\exists b < p) \\ &\quad \mathrm{EndSequent}(p) = \langle g, d \frown \langle {}^{\#}({}^{\#} \frown a \frown {}^{\#}\land^{\#} \frown b \frown {}^{\#})^{\#}\rangle\rangle \land \\ &\quad \mathrm{EndSequent}((p)_1) = \langle g, d \frown \langle a\rangle\rangle \land \\ &\quad \mathrm{EndSequent}((p)_2) = \langle g, d \frown \langle b\rangle\rangle \land \\ &\quad (p)_0 = 2 \land \mathrm{LastRule}(p) = 10.\end{aligned}$$

The individual lines express, respectively, "there is a sequence (Γ) with Gödel number g, there is a sequence (Δ) with Gödel number d, a formula (φ) with Gödel number a, and a formula (ψ) with Gödel number b," such that "the end-sequent of π is $\Gamma \Rightarrow \Delta, \varphi \land \psi$," "the end-sequent of π_1 is $\Gamma \Rightarrow \Delta, \varphi$," "the end-sequent of π_2 is $\Gamma \Rightarrow \Delta, \psi$," and "$\pi$ has two immediate subderivations and the last inference rule is $\land$R (with number 10)."

The last inference in π is a correct application of $\exists$R iff there are sequences Γ and Δ, a formula φ, a variable x, and a term t, such that the end-sequent

of π is $\Gamma \Rightarrow \Delta, \exists x\, \varphi$ and the end-sequent of π_1 is $\Gamma \Rightarrow \Delta, \varphi[t/x]$. So in terms of Gödel numbers, we have $\mathrm{FollowsBy}_{\exists \mathrm{R}}(p)$ iff

$$\begin{aligned}
&(\exists g < p)\ (\exists d < p)\ (\exists a < p)\ (\exists x < p)\ (\exists t < p)\\
&\qquad \mathrm{EndSequent}(p) = \langle g, d \frown \langle {}^{\#}\exists^{\#} \frown x \frown a\rangle\rangle \land\\
&\qquad \mathrm{EndSequent}((p)_1) = \langle g, d \frown \langle \mathrm{Subst}(a, t, x)\rangle\rangle \land\\
&\qquad (p)_0 = 1 \land \mathrm{LastRule}(p) = 18.
\end{aligned}$$

We then define $\mathrm{Correct}(p)$ as

$$\begin{aligned}
&\mathrm{Sequent}(\mathrm{EndSequent}(p)) \land\\
&\qquad [(\mathrm{LastRule}(p) = 1 \land \mathrm{FollowsBy}_{\mathrm{WL}}(p)) \lor \cdots \lor\\
&\qquad\quad (\mathrm{LastRule}(p) = 20 \land \mathrm{FollowsBy}_{=}(p)) \lor\\
&\qquad\qquad\qquad (p)_0 = 0 \land \mathrm{InitialSeq}(\mathrm{EndSequent}(p))]
\end{aligned}$$

The first line ensures that the end-sequent of d is actually a sequent consisting of sentences. The last line covers the case where p is just an initial sequent. □

Problem art.5. Define the following properties as in Proposition art.15:

1. $\mathrm{FollowsBy}_{\mathrm{Cut}}(p)$,
2. $\mathrm{FollowsBy}_{\to \mathrm{L}}(p)$,
3. $\mathrm{FollowsBy}_{=}(p)$,
4. $\mathrm{FollowsBy}_{\forall \mathrm{R}}(p)$.

For the last one, you will have to also show that you can test primitive recursively if the last inference of the derivation with Gödel number p satisfies the eigenvariable condition, i.e., the eigenvariable a of the $\forall$R does not occur in the end-sequent.

Proposition art.16. *The relation* $\mathrm{Deriv}(p)$ *which holds if* p *is the Gödel number of a correct derivation* π*, is primitive recursive.* *inc:art:plk:prop:deriv*

Proof. A derivation π is correct if every one of its inferences is a correct application of a rule, i.e., if every one of its sub-derivations ends in a correct inference. So, $\mathrm{Deriv}(d)$ iff

$$(\forall i < \mathrm{len}(\mathrm{SubtreeSeq}(p)))\ \mathrm{Correct}((\mathrm{SubtreeSeq}(p))_i. \qquad \square$$

Proposition art.17. *Suppose* Γ *is a primitive recursive set of sentences. Then the relation* $\mathrm{Prf}_\Gamma(x, y)$ *expressing "*x *is the code of a derivation* π *of* $\Gamma_0 \Rightarrow \varphi$ *for some finite* $\Gamma_0 \subseteq \Gamma$ *and* y *is the Gödel number of* φ*" is primitive recursive.*

Proof. Suppose "$y \in \Gamma$" is given by the primitive recursive predicate $R_\Gamma(y)$. We have to show that $\mathrm{Prf}_\Gamma(x, y)$ which holds iff y is the Gödel number of a sentence φ and x is the code of an **LK**-derivation with end-sequent $\Gamma_0 \Rightarrow \varphi$ is primitive recursive.

By the previous proposition, the property $\mathrm{Deriv}(x)$ which holds iff x is the code of a correct derivation π in **LK** is primitive recursive. If x is such a code, then $\mathrm{EndSequent}(x)$ is the code of the end-sequent of π, and so $(\mathrm{EndSequent}(x))_0$ is the code of the left side of the end sequent and $(\mathrm{EndSequent}(x))_1$ the right side. So we can express "the right side of the end-sequent of π is φ" as $\mathrm{len}((\mathrm{EndSequent}(x))_1) = 1 \land ((\mathrm{EndSequent}(x))_1)_0 = x$. The left side of the end-sequent of π is of course automatically finite, we just have to express that every sentence in it is in Γ. Thus we can define $\mathrm{Prf}_\Gamma(x, y)$ by

$$\begin{aligned}\mathrm{Prf}_\Gamma(x, y) \Leftrightarrow \mathrm{Deriv}(x) \land & \\ & (\forall i < \mathrm{len}((\mathrm{EndSequent}(x))_0))\ R_\Gamma(((\mathrm{EndSequent}(x))_0)_i) \land \\ & \mathrm{len}((\mathrm{EndSequent}(x))_1) = 1 \land ((\mathrm{EndSequent}(x))_1)_0 = y. \qquad \square\end{aligned}$$

art.7 Derivations in Natural Deduction

inc:art:pnd:sec In order to arithmetize derivations, we must represent derivations as numbers. explanation Since derivations are trees of formulas where each inference carries one or two labels, a recursive representation is the most obvious approach: we represent a derivation as a tuple, the components of which are the number of immediate sub-derivations leading to the premises of the last inference, the representations of these sub-derivations, and the end-formula, the discharge label of the last inference, and a number indicating the type of the last inference.

Definition art.18. If δ is a derivation in natural deduction, then ${}^{\#}\delta^{\#}$ is defined inductively as follows:

1. If δ consists only of the assumption φ, then ${}^{\#}\delta^{\#}$ is $\langle 0, {}^{\#}\varphi^{\#}, n\rangle$. The number n is 0 if it is an undischarged assumption, and the numerical label otherwise.

2. If δ ends in an inference with one, two, or three premises, then ${}^{\#}\delta^{\#}$ is

$$\begin{aligned}&\langle 1, {}^{\#}\delta_1{}^{\#}, {}^{\#}\varphi^{\#}, n, k\rangle, \\ &\langle 2, {}^{\#}\delta_1{}^{\#}, {}^{\#}\delta_2{}^{\#}, {}^{\#}\varphi^{\#}, n, k\rangle, \text{ or} \\ &\langle 3, {}^{\#}\delta_1{}^{\#}, {}^{\#}\delta_2{}^{\#}, {}^{\#}\delta_3{}^{\#}, {}^{\#}\varphi^{\#}, n, k\rangle,\end{aligned}$$

   respectively. Here δ_1, δ_2, δ_3 are the sub-derivations ending in the premise(s) of the last inference in δ, φ is the conclusion of the last inference in δ, n is the discharge label of the last inference (0 if the inference does not discharge any assumptions), and k is given by the following table according to which rule was used in the last inference.

Rule:	$\wedge$Intro	$\wedge$Elim	$\vee$Intro	$\vee$Elim
k:	1	2	3	4
Rule:	$\rightarrow$Intro	$\rightarrow$Elim	$\neg$Intro	$\neg$Elim
k:	5	6	7	8
Rule:	$\bot_I$	$\bot_C$	$\forall$Intro	$\forall$Elim
k:	9	10	11	12
Rule:	$\exists$Intro	$\exists$Elim	=Intro	=Elim
k:	13	14	15	16

Example art.19. Consider the very simple derivation

$$1\,\dfrac{\dfrac{[\varphi \wedge \psi]^1}{\varphi}\,\wedge\text{Elim}}{(\varphi \wedge \psi) \rightarrow \varphi}\,\rightarrow\text{Intro}$$

The Gödel number of the assumption would be $d_0 = \langle 0, {}^{\#}\varphi \wedge \psi^{\#}, 1\rangle$. The Gödel number of the derivation ending in the conclusion of $\wedge$Elim would be $d_1 = \langle 1, d_0, {}^{\#}\varphi^{\#}, 0, 2\rangle$ (1 since $\wedge$Elim has one premise, the Gödel number of conclusion φ, 0 because no assumption is discharged, and 2 is the number coding $\wedge$Elim). The Gödel number of the entire derivation then is $\langle 1, d_1, {}^{\#}((\varphi \wedge \psi) \rightarrow \varphi)^{\#}, 1, 5\rangle$, i.e.,

$$\langle 1, \langle 1, \langle 0, {}^{\#}(\varphi \wedge \psi)^{\#}, 1\rangle, {}^{\#}\varphi^{\#}, 0, 2\rangle, {}^{\#}((\varphi \wedge \psi) \rightarrow \varphi)^{\#}, 1, 5\rangle.$$

explanation

Having settled on a representation of derivations, we must also show that we can manipulate Gödel numbers of such derivations primitive recursively, and express their essential properties and relations. Some operations are simple: e.g., given a Gödel number d of a derivation, $\mathrm{EndFmla}(d) = (d)_{(d)_0+1}$ gives us the Gödel number of its end-formula, $\mathrm{DischargeLabel}(d) = (d)_{(d)_0+2}$ gives us the discharge label and $\mathrm{LastRule}(d) = (d)_{(d)_0+3}$ the number indicating the type of the last inference. Some are much harder. We'll at least sketch how to do this. The goal is to show that the relation "δ is a derivation of φ from Γ" is a primitive recursive relation of the Gödel numbers of δ and φ.

Proposition art.20. *The following relations are primitive recursive:*

1. *φ occurs as an assumption in δ with label n.*
2. *All assumptions in δ with label n are of the form φ (i.e., we can discharge the assumption φ using label n in δ).*

Proof. We have to show that the corresponding relations between Gödel numbers of formulas and Gödel numbers of derivations are primitive recursive.

1. We want to show that $\mathrm{Assum}(x, d, n)$, which holds if x is the Gödel number of an assumption of the derivation with Gödel number d labelled n,

is primitive recursive. This is the case if the derivation with Gödel number $\langle 0, x, n\rangle$ is a sub-derivation of d. Note that the way we code derivations is a special case of the coding of trees introduced in **??**, so the primitive recursive function $\mathrm{SubtreeSeq}(d)$ gives a sequence of Gödel numbers of all sub-derivations of d (of length a most d). So we can define

$$\mathrm{Assum}(x, d, n) \Leftrightarrow (\exists i < d)\ (\mathrm{SubtreeSeq}(d))_i = \langle 0, x, n\rangle.$$

2. We want to show that $\mathrm{Discharge}(x, d, n)$, which holds if all assumptions with label n in the derivation with Gödel number d all are the formula with Gödel number x. But this relation holds iff $(\forall y < d)\ (\mathrm{Assum}(y, d, n) \rightarrow y = x)$. □

inc:art:pnd:prop:followsby **Proposition art.21.** *The property* $\mathrm{Correct}(d)$ *which holds iff the last inference in the derivation* δ *with Gödel number* d *is correct, is primitive recursive.*

Proof. Here we have to show that for each rule of inference R the relation $\mathrm{FollowsBy}_R(d)$ is primitive recursive, where $\mathrm{FollowsBy}_R(d)$ holds iff d is the Gödel number of derivation δ, and the end-formula of δ follows by a correct application of R from the immediate sub-derivations of δ.

A simple case is that of the $\wedge$Intro rule. If δ ends in a correct $\wedge$Intro inference, it looks like this:

$$\dfrac{\begin{array}{c}\vdots\ \delta_1\\ \varphi\end{array} \qquad \begin{array}{c}\vdots\ \delta_2\\ \psi\end{array}}{\varphi \wedge \psi}\ \wedge\mathrm{Intro}$$

Then the Gödel number d of δ is $\langle 2, d_1, d_2, {}^{\#}(\varphi \wedge \psi)^{\#}, 0, k\rangle$ where $\mathrm{EndFmla}(d_1) = {}^{\#}\varphi^{\#}$, $\mathrm{EndFmla}(d_2) = {}^{\#}\psi^{\#}$, $n = 0$, and $k = 1$. So we can define $\mathrm{FollowsBy}_{\wedge\mathrm{Intro}}(d)$ as

$$\begin{aligned}&(d)_0 = 2 \wedge \mathrm{DischargeLabel}(d) = 0 \wedge \mathrm{LastRule}(d) = 1 \wedge\\ &\mathrm{EndFmla}(d) = {}^{\#}(^{\#} \frown \mathrm{EndFmla}((d)_1) \frown {}^{\#}\wedge^{\#} \frown \mathrm{EndFmla}((d)_2) \frown {}^{\#})^{\#}.\end{aligned}$$

Another simple example if the =Intro rule. Here the premise is an empty derivation, i.e., $(d)_1 = 0$, and no discharge label, i.e., $n = 0$. However, φ must be of the form $t = t$, for a closed term t. Here, a primitive recursive definition is

$$\begin{aligned}&(d)_0 = 1 \wedge (d)_1 = 0 \wedge \mathrm{DischargeLabel}(d) = 0 \wedge\\ &\qquad (\exists t < d)\ (\mathrm{ClTerm}(t) \wedge \mathrm{EndFmla}(d) = {}^{\#}{=}(^{\#} \frown t \frown {}^{\#},^{\#} \frown t \frown {}^{\#})^{\#})\end{aligned}$$

For a more complicated example, $\mathrm{FollowsBy}_{\rightarrow\mathrm{Intro}}(d)$ holds iff the end-formula of δ is of the form $(\varphi \rightarrow \psi)$, where the end-formula of δ_1 is ψ, and

any assumption in δ labelled n is of the form φ. We can express this primitive recursively by

$$\begin{aligned}&(d)_0 = 1 \land \\ &\qquad (\exists a < d)\ (\mathrm{Discharge}(a, (d)_1, \mathrm{DischargeLabel}(d)) \land \\ &\qquad \mathrm{EndFmla}(d) = ({}^{\#}(^{\#} \frown a \frown {}^{\#}{\to}^{\#} \frown \mathrm{EndFmla}((d)_1) \frown {}^{\#})^{\#}))\end{aligned}$$

(Think of a as the Gödel number of φ).

For another example, consider $\exists$Intro. Here, the last inference in δ is correct iff there is a formula φ, a closed term t and a variable x such that $\varphi[t/x]$ is the end-formula of the derivation δ_1 and $\exists x\, \varphi$ is the conclusion of the last inference. So, $\mathrm{FollowsBy}_{\exists\mathrm{Intro}}(d)$ holds iff

$$\begin{aligned}&(d)_0 = 1 \land \mathrm{DischargeLabel}(d) = 0 \land \\ &\qquad (\exists a < d)\ (\exists x < d)\ (\exists t < d)\ (\mathrm{ClTerm}(t) \land \mathrm{Var}(x) \land \\ &\qquad \mathrm{Subst}(a, t, x) = \mathrm{EndFmla}((d)_1) \land \mathrm{EndFmla}(d) = ({}^{\#}\exists^{\#} \frown x \frown a)).\end{aligned}$$

We then define $\mathrm{Correct}(d)$ as

$$\begin{aligned}&\mathrm{Sent}(\mathrm{EndFmla}(d)) \land \\ &\qquad (\mathrm{LastRule}(d) = 1 \land \mathrm{FollowsBy}_{\land\mathrm{Intro}}(d)) \lor \cdots \lor \\ &\qquad (\mathrm{LastRule}(d) = 16 \land \mathrm{FollowsBy}_{=\mathrm{Elim}}(d)) \lor \\ &\qquad\qquad (\exists n < d)\ (\exists x < d)\ (d = \langle 0, x, n\rangle).\end{aligned}$$

The first line ensures that the end-formula of d is a sentence. The last line covers the case where d is just an assumption. □

Problem art.6. Define the following properties as in Proposition art.21:

1. $\mathrm{FollowsBy}_{\to\mathrm{Elim}}(d)$,
2. $\mathrm{FollowsBy}_{=\mathrm{Elim}}(d)$,
3. $\mathrm{FollowsBy}_{\lor\mathrm{Elim}}(d)$,
4. $\mathrm{FollowsBy}_{\forall\mathrm{Intro}}(d)$.

For the last one, you will have to also show that you can test primitive recursively if the last inference of the derivation with Gödel number d satisfies the eigenvariable condition, i.e., the eigenvariable a of the $\forall$Intro inference occurs neither in the end-formula of d nor in an open assumption of d. You may use the primitive recursive predicate OpenAssum from Proposition art.23 for this.

Proposition art.22. *The relation* $\mathrm{Deriv}(d)$ *which holds if* d *is the Gödel number of a correct derivation* δ*, is primitive recursive.* *inc:art:pnd:prop:deriv*

Proof. A derivation δ is correct if every one of its inferences is a correct application of a rule, i.e., if every one of its sub-derivations ends in a correct inference. So, $\mathrm{Deriv}(d)$ iff

$$(\forall i < \mathrm{len}(\mathrm{SubtreeSeq}(d)))\ \mathrm{Correct}((\mathrm{SubtreeSeq}(d))_i) \qquad \square$$

inc:art:pnd:prop:openassum

Proposition art.23. *The relation* $\mathrm{OpenAssum}(z, d)$ *that holds if* z *is the Gödel number of an undischarged assumption* φ *of the derivation* δ *with Gödel number* d*, is primitive recursive.*

Proof. An occurrence of an assumption is discharged if it occurs with label n in a sub-derivation of δ that ends in a rule with discharge label n. So φ is an undischarged assumption of δ if at least one of its occurrences is not discharged in δ. We must be careful: δ may contain both discharged and undischarged occurrences of φ.

Consider a sequence $\delta_0, \ldots, \delta_k$ where $\delta_0 = \delta$, δ_k is the assumption $[\varphi]^n$ (for some n), and δ_{i+1} is an immediate sub-derivation of δ_i. If such a sequence exists in which no δ_i ends in an inference with discharge label n, then φ is an undischarged assumption of δ.

The primitive recursive function $\mathrm{SubtreeSeq}(d)$ provides us with a sequence of Gödel numbers of all sub-derivations of δ. Any sequence of Gödel numbers of sub-derivations of δ is a subsequence of it. Being a subsequence of is a primitive recursive relation: $\mathrm{Subseq}(s, s')$ holds iff $(\forall i < \mathrm{len}(s))\ \exists j < \mathrm{len}(s')\ (s)_i = (s)_j$. Being an immediate sub-derivation is as well: $\mathrm{Subderiv}(d, d')$ iff $(\exists j < (d')_0)\ d = (d')_j$. So we can define $\mathrm{OpenAssum}(z, d)$ by

$$\begin{aligned}(\exists s < \mathrm{SubtreeSeq}(d))\ (\mathrm{Subseq}(s, \mathrm{SubtreeSeq}(d)) \land (s)_0 = d \land \\ (\exists n < d)\ ((s)_{\mathrm{len}(s) \dot{-} 1} = \langle 0, z, n\rangle \land \\ (\forall i < (\mathrm{len}(s) \dot{-} 1))\ (\mathrm{Subderiv}((s)_{i+1}, (s)_i)] \land \\ \mathrm{DischargeLabel}((s)_{i+1}) \neq n))). \qquad \square\end{aligned}$$

Proposition art.24. *Suppose* Γ *is a primitive recursive set of sentences. Then the relation* $\mathrm{Prf}_\Gamma(x, y)$ *expressing "*x *is the code of a derivation* δ *of* φ *from undischarged assumptions in* Γ *and* y *is the Gödel number of* φ*" is primitive recursive.*

Proof. Suppose "$y \in \Gamma$" is given by the primitive recursive predicate $R_\Gamma(y)$. We have to show that $\mathrm{Prf}_\Gamma(x, y)$ which holds iff y is the Gödel number of a sentence φ and x is the code of a natural deduction derivation with end formula φ and all undischarged assumptions in Γ is primitive recursive.

By Proposition art.22, the property $\mathrm{Deriv}(x)$ which holds iff x is the Gödel number of a correct derivation δ in natural deduction is primitive recursive. Thus we can define $\mathrm{Prf}_\Gamma(x, y)$ by

$$\begin{aligned}\mathrm{Prf}_\Gamma(x, y) \Leftrightarrow \mathrm{Deriv}(x) \land \mathrm{EndFmla}(x) = y \land \\ (\forall z < x)\ (\mathrm{OpenAssum}(z, x) \to R_\Gamma(z)). \qquad \square\end{aligned}$$

art.8 Axiomatic Derivations

explanation In order to arithmetize axiomatic derivations, we must represent derivations as numbers. Since derivations are simply sequences of formulas, the obvious approach is to code every derivation as the code of the sequence of codes of formulas in it. inc:art:pax:sec

Definition art.25. If δ is an axiomatic derivation consisting of formulas φ_1, $\dots$, φ_n, then ${}^{\#}\delta^{\#}$ is

$$\langle {}^{\#}\varphi_1{}^{\#}, \dots, {}^{\#}\varphi_n{}^{\#} \rangle.$$

Example art.26. Consider the very simple derivation:

1. $\psi \to (\psi \lor \varphi)$
2. $(\psi \to (\psi \lor \varphi)) \to (\varphi \to (\psi \to (\psi \lor \varphi)))$
3. $\varphi \to (\psi \to (\psi \lor \varphi))$

The Gödel number of this derivation would be

$$\begin{aligned} \langle\, & {}^{\#}\psi \to (\psi \lor \varphi)^{\#}, \\ & {}^{\#}(\psi \to (\psi \lor \varphi)) \to (\varphi \to (\psi \to (\psi \lor \varphi)))^{\#}, \\ & {}^{\#}\varphi \to (\psi \to (\psi \lor \varphi))^{\#} \,\rangle. \end{aligned}$$

explanation Having settled on a representation of derivations, we must also show that we can manipulate such derivations primitive recursively, and express their essential properties and relations so. Some operations are simple: e.g., given a Gödel number d of a derivation, $(d)_{\mathrm{len}(d)-1}$ gives us the Gödel number of its end-formula. Some are much harder. We'll at least sketch how to do this. The goal is to show that the relation "δ is a derivation of φ from Γ" is primitive recursive in the Gödel numbers of δ and φ.

Proposition art.27. *The following relations are primitive recursive:* *inc:art:pax:prop:followsby*

1. *φ is an axiom.*
2. *The i-th line in δ is justified by modus ponens*
3. *The i-th line in δ is justified by* QR.
4. *δ is a correct derivation.*

Proof. We have to show that the corresponding relations between Gödel numbers of formulas and Gödel numbers of derivations are primitive recursive.

1. We have a given list of axiom schemas, and φ is an axiom if it is of the form given by one of these schemas. Since the list of schemas is finite, it suffices to show that we can test primitive recursively, for each axiom schema, if φ is of that form. For instance, consider the axiom schema

$$\psi \to (\chi \to \psi).$$

φ is an instance of this axiom schema if there are formulas ψ and χ such that we obtain φ when we concatenate '(' with ψ with '$\rightarrow$' with '(' with χ with '$\rightarrow$' with ψ and with ')'. We can test the corresponding property of the Gödel number n of φ, since concatenation of sequences is primitive recursive and the Gödel numbers of ψ and χ must be smaller than the Gödel number of φ, since when the relation holds, both ψ and χ are sub-formulas of φ. Hence, we can define:

$$\begin{aligned}\mathrm{IsAx}_{\psi\rightarrow(\chi\rightarrow\psi)}(n) \Leftrightarrow (\exists b < n)\ (\exists c < n)\ (\mathrm{Sent}(b) \land \mathrm{Sent}(c) \land \\ n = {}^{\#}(^{\#} \frown b \frown {}^{\#}\rightarrow^{\#} \frown {}^{\#}(^{\#} \frown c \frown {}^{\#}\rightarrow^{\#} \frown b \frown {}^{\#}))^{\#}).\end{aligned}$$

If we have such a definition for each axiom schema, their disjunction defines the property $\mathrm{IsAx}(n)$, "n is the Gödel number of an axiom."

2. The i-th line in δ is justified by modus ponens iff there are lines j and $k < i$ where the sentence on line j is some formula φ, the sentence on line k is $\varphi \rightarrow \psi$, and the sentence on line i is ψ.

$$\begin{aligned}\mathrm{MP}(d, i) \Leftrightarrow (\exists j < i)\ (\exists k < i) \\ (d)_k = {}^{\#}(^{\#} \frown (d)_j \frown {}^{\#}\rightarrow^{\#} \frown (d)_i \frown {}^{\#})^{\#}\end{aligned}$$

Since bounded quantification, concatenation, and $=$ are primitive recursive, this defines a primitive recursive relation.

3. A line in δ is justified by QR if it is of the form $\psi \rightarrow \forall x\, \varphi(x)$, a preceding line is $\psi \rightarrow \varphi(c)$ for some constant symbol c, and c does on occur in ψ. This is the case iff

   a) there is a sentence ψ and
   b) a formula $\varphi(x)$ with a single variable x free so that
   c) line i contains $\psi \rightarrow \forall x\, \varphi(x)$
   d) some line $j < i$ contains $\psi \rightarrow \varphi[c/x]$ for a constant c
   e) which does not occur in ψ.

   All of these can be tested primitive recursively, since the Gödel numbers of ψ, $\varphi(x)$, and x are less than the Gödel number of the formula on line i, and that of a less than the Gödel number of the formula on line j:

$$\begin{aligned}\mathrm{QR}_1(d, i) \Leftrightarrow (\exists b < (d)_i)\ (\exists x < (d)_i)\ (\exists a < (d)_i)\ (\exists c < (d)_j)\ (\\ \mathrm{Var}(x) \land \mathrm{Const}(c) \land \\ (d)_i = {}^{\#}(^{\#} \frown b \frown {}^{\#}\rightarrow^{\#} \frown {}^{\#}\forall^{\#} \frown x \frown a \frown {}^{\#})^{\#} \land \\ (d)_j = {}^{\#}(^{\#} \frown b \frown {}^{\#}\rightarrow^{\#} \frown \mathrm{Subst}(a, c, x) \frown {}^{\#})^{\#} \land \\ \mathrm{Sent}(b) \land \mathrm{Sent}(\mathrm{Subst}(a, c, x)) \land (\forall k < \mathrm{len}(b))\ (b)_k \neq (c)_0)\end{aligned}$$

   Here we assume that c and x are the Gödel numbers of the variable and constant considered as terms (i.e., not their symbol codes). We test that

x is the only free variable of $\varphi(x)$ by testing if $\varphi(x)[c/x]$ is a sentence, and ensure that c does not occur in ψ by requiring that every symbol of ψ is different from c.

We leave the other version of QR as an exercise.

4. d is the Gödel number of a correct derivation iff every line in it is an axiom, or justified by modus ponens or QR. Hence:

$$\mathrm{Deriv}(d) \Leftrightarrow (\forall i < \mathrm{len}(d))\ (\mathrm{IsAx}((d)_i) \lor \mathrm{MP}(d,i) \lor \mathrm{QR}(d,i)) \qquad \square$$

Problem art.7. Define the following relations as in Proposition art.27:

1. $\mathrm{IsAx}_{\varphi\to(\psi\to(\varphi\land\psi))}(n)$,
2. $\mathrm{IsAx}_{\forall x\,\varphi(x)\to\varphi(t)}(n)$,
3. $\mathrm{QR}_2(d,i)$ (for the other version of QR).

Proposition art.28. *Suppose Γ is a primitive recursive set of sentences. Then the relation $\mathrm{Prf}_\Gamma(x,y)$ expressing "x is the code of a derivation δ of φ from Γ and y is the Gödel number of φ" is primitive recursive.*

Proof. Suppose "$y \in \Gamma$" is given by the primitive recursive predicate $R_\Gamma(y)$. We have to show that the relation $\mathrm{Prf}_\Gamma(x,y)$ is primitive recursive, where $\mathrm{Prf}_\Gamma(x,y)$ holds iff y is the Gödel number of a sentence φ and x is the code of a derivation of φ from Γ.

By the previous proposition, the property $\mathrm{Deriv}(x)$ which holds iff x is the code of a correct derivation δ is primitive recursive. However, that definition did not take into account the set Γ as an additional way to justify lines in the derivation. Our primitive recursive test of whether a line is justified by QR also left out of consideration the requirement that the constant c is not allowed to occur in Γ. It is possible to amend our definition so that it takes into account Γ directly, but it is easier to use Deriv and the deduction theorem. $\Gamma \vdash \varphi$ iff there is some finite list of sentences $\psi_1, \dots, \psi_n \in \Gamma$ such that $\{\psi_1, \dots, \psi_n\} \vdash \varphi$. And by the deduction theorem, this is the case if $\vdash (\psi_1 \to (\psi_2 \to \cdots (\psi_n \to \varphi) \cdots))$. Whether a sentence with Gödel number z is of this form can be tested primitive recursively. So, instead of considering x as the Gödel number of a derivation of the sentence with Gödel number y *from* Γ, we consider x as the Gödel number of a derivation of a nested conditional of the above form from $\emptyset$.

First, if we have a sequence of sentences, we can primitive recursively form the conditional with all these sentences as antecedents and given sentence as consequent:

$$\begin{aligned}
\mathrm{hCond}(s,y,0) &= y \\
\mathrm{hCond}(s,y,n+1) &= {}^{\#}(^{\#} \frown (s)_n \frown {}^{\#}{\to}^{\#} \frown \mathrm{Cond}(s,y,n) \frown {}^{\#})^{\#} \\
\mathrm{Cond}(s,y) &= \mathrm{hCond}(s,y,\mathrm{len}(s))
\end{aligned}$$

So we can define $\mathrm{Prf}_{\Gamma}(x, y)$ by

$$\begin{aligned}\mathrm{Prf}_{\Gamma}(x, y) \Leftrightarrow (\exists s < \mathrm{sequenceBound}(x, x))\ (&\\ &(x)_{\mathrm{len}(x)-1} = \mathrm{Cond}(s, y) \wedge \\ &(\forall i < \mathrm{len}(s))\ (s)_i \in \Gamma \wedge \\ &\mathrm{Deriv}(x)).\end{aligned}$$

The bound on s is given by considering that each $(s)_i$ is the Gödel number of a sub-formula of the last line of the derivation, i.e., is less than $(x)_{\mathrm{len}(x)-1}$. The number of antecedents $\psi \in \Gamma$, i.e., the length of s, is less than the length of the last line of x. $\square$

Photo Credits

Bibliography