Chapter udf

Arithmetization of Syntax

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 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 usign computable func-

tions. 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, of being the Gödel number of a corret 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 assignes a Gödel number to $[t/x]$, say, n. This mapping—of k, l, 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 prower 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

inc:art:cod:sec

The basic language $\mathcal{L}$ of first order logic makes use of the symbols

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

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 assigned 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 num-

bers. 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 $\langle n_0, \ldots, n_k \rangle$ is the number coding the sequence of numbers n_0, …, 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:

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

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^n, 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 $s_0, \ldots, s_{n-1}$ is a sequence of symbols, its *Gödel number* is $\langle \mathrm{c}_{s_0}, \ldots, \mathrm{c}_{s_{n-1}}\rangle$.

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* $\ulcorner v_5 \urcorner$ 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, …, k_{n-1} is a sequence of numbers, then the code of the sequence $\langle k_0, \ldots, k_{n-1}\rangle$ in the power-of-primes coding is

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

where p_i is the i-th prime (starting with $p_0 = 2$). So for instance, the formula $v_0 = \mathsf{0}$, 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, $c_=$ 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 $\ulcorner = (v_0, c_0)\urcorner$ is

$$2^{c_=+1} \cdot 3^{c_(+1} \cdot 5^{c_{v_0}+1} \cdot 7^{c_,+1} \cdot 11^{c_{c_0}+1} \cdot 13^{c_)+1} =$$
$$2^{2^1\cdot 3^8+1} \cdot 3^{2^1\cdot 3^9+1} \cdot 5^{2^2\cdot 3^1+1} \cdot 7^{2^1\cdot 3^{11}+1} \cdot 11^{2^3\cdot 3^1+1} \cdot 13^{2^1\cdot 3^{10}+1} =$$
$$2^{13\,123} \cdot 3^{39\,367} \cdot 5^{13} \cdot 7^{354\,295} \cdot 11^{25} \cdot 13^{118\,099}.$$

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

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^n_j.

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., $y = \langle \ulcorner s_0 \urcorner, \ldots, \ulcorner s_k \urcorner\rangle$. It codes a formation sequence for the term with Gödel number x iff for all $i < k$:

1. there is a j such that $(y)_i = \ulcorner v_j \urcorner$, or
2. there is a j such that $(y)_i = \ulcorner c_j \urcorner$, 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

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

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

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

Alternative proof of Proposition art.5. The inductive definition says that constant symbols and variables are terms, and if t_1, ..., t_n are terms, then so is $f_j^n(t_1, \ldots, t_n)$, for any n and j. So terms are formed in stages: constant symbols and variables at stage 0, terms involving one function symbol at stage 1, those involving at least two nested function symbols at stage 2, etc. Let's say that a sequence of symbols s is a term of level l iff s can be formed by applying the inductive definition of terms l (or fewer) times, i.e., it "becomes" a term by stage l or before. So s is a term of level $l + 1$ iff

1. s is a variable v_j, or
2. s is a constant symbol c_j, or
3. s is built from n terms t_1, ..., t_n of level l and 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.

The number x is the Gödel number of a term s of level $l + 1$ iff

1. there is a j such that $x = {}^{\ulcorner}v_j{}^{\urcorner}$, or
2. there is a j such that $x = {}^{\ulcorner}c_j{}^{\urcorner}$, or
3. there is an n, a j, and a number $z = \langle z_1, \ldots, z_n\rangle$ such that each z_i is the Gödel number of a term of level l and

$$x = {}^{\ulcorner}f_j^n({}^{\urcorner} \frown \mathrm{flatten}(z) \frown {}^{\ulcorner}){}^{\urcorner},$$

and moreover $(y)_{k-1} = x$.

The indices j, n, the Gödel numbers z_i of the terms t_i, and the code z of the sequence $\langle z_1, \ldots, z_n \rangle$, in (3) are all less than x. So we get a primitive recursive definition by:

$$\begin{aligned}
\mathrm{lTerm}(x, 0) &= \mathrm{Var}(x) \vee \mathrm{Const}(x) \\
\mathrm{lTerm}(x, l+1) &= \mathrm{Var}(x) \vee \mathrm{Const}(x) \vee \\
&\qquad (\exists z < x)\ ((\forall i < \mathrm{len}(z))\ \mathrm{lTerm}((z)_i, l) \wedge \\
&\qquad\quad (\exists j < x)\ x = ({}^{\dot{\frown}} f_j^{\mathrm{len}(z)} ({}^{\#} \mathbin{\dot{\frown}} \mathrm{flatten}(z) \mathbin{\dot{\frown}} {}^{\dot{\frown}})^{\#}))
\end{aligned}$$

We can now define $\mathrm{Term}(x)$ by $\mathrm{lTerm}(x, x)$, since the level of a term is always less than the Gödel number of the term. □

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

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

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

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

□

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 $x =$
$$ {}^{\dot{\frown}} P_j^n ({}^{\#} \mathbin{\dot{\frown}} \mathrm{flatten}(z) \mathbin{\dot{\frown}} {}^{\dot{\frown}})^{\#}. $$
2. There are $z_1, z_2 < x$ such that $\mathrm{Term}(z_1)$, $\mathrm{Term}(z_2)$, and $x =$
$$ {}^{\dot{\frown}}{=}({}^{\#} \mathbin{\dot{\frown}} z_1 \mathbin{\dot{\frown}} {}^{\dot{\frown}},{}^{\#} \mathbin{\dot{\frown}} z_2 \mathbin{\dot{\frown}} {}^{\dot{\frown}})^{\#}. $$
3. $x = {}^{\dot{\frown}}\bot^{\#}$.
4. $x = {}^{\dot{\frown}}\top^{\#}$.

□

inc:art:frm: prop:frm-primrec

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

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

Problem art.3. Give a detailed proof of Proposition art.8 along the lines of the alternate proof of Proposition art.5

inc:art:frm: prop:freeocc-primrec

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.*

Proof. Exercise. □

Problem art.4. 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

$$(\forall i < \mathrm{len}(x))\ (\forall z < x)\ ((\exists j < z)\ z = {}^{\ulcorner}v_j{}^{\urcorner} \rightarrow \neg\mathrm{FreeOcc}(x, z, i)).$$

□

art.5 Substitution

inc:art:sub: sec

inc:art:sub: prop:subst-primrec

Proposition art.11. *There is a primitive recursive function* $\mathrm{Subst}(x, y, z)$ *with the property that*

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

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

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

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

Proposition art.12. *The relation* 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.* inc:art:sub:prop:free-for

Proof. Exercise. □

Problem art.5. Prove Proposition art.12

art.6 Derivations in LK

explanation In order to arithmetize derivations, we must represent derivations as numbers. 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. inc:art:plk:sec

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

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

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

If π is a derivation in **LK**, then ${}^{\ulcorner}\pi^{\urcorner}$ is

1. $\langle 0, {}^{\ulcorner}\Gamma \Rightarrow \Delta^{\urcorner} \rangle$ if π consists only of the initial sequent $\Gamma \Rightarrow \Delta$.

2. $\langle 1, {}^{\ulcorner}\Gamma \Rightarrow \Delta^{\urcorner}, k, {}^{\ulcorner}\pi'^{\urcorner} \rangle$ if π ends in an inference with one premise, k is given by the following table according to which rule was used in the last inference, and π' is the immediate subproof ending in the premise of the last inference.

Rule:	WL	WR	CL	CR	XL	XR
k:	1	2	3	4	5	6

Rule:	¬L	¬R	∧L	∨R	→R
k:	7	8	9	10	11

Rule:	∀L	∀R	∃L	∃R	=
k:	12	13	14	15	16

3. $\langle 2, {}^{\ulcorner}\Gamma \Rightarrow \Delta^{\urcorner}, k, {}^{\ulcorner}\pi'^{\urcorner}, {}^{\ulcorner}\pi''^{\urcorner} \rangle$ if π ends in an inference with two premises, k is given by the following table according to which rule was used in the last inference, and π', π'' are the immediate subproof ending in the left and right premise of the last inference, respectively.

Rule:	Cut	∧R	∨L	→L
k:	1	2	3	4

explanation

Having settled on a representation of derivations, we must also show that we can manipulate such derivations primite recursively, and express their essential properties and relations so. Some operations are simple: e.g., given a Gödel number d of a derivation, $(s)_1$ gives us the Gödel number of its end-sequent. 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.14. *The following relations are primitive recursive:*

1. *$\Gamma \Rightarrow \Delta$ is an initial sequent.*
2. *$\Gamma \Rightarrow \Delta$ follows from $\Gamma' \Rightarrow \Delta'$ (and $\Gamma'' \Rightarrow \Delta''$) by a rule of* **LK**.
3. *π is a correct* **LK***-derivation.*

Proof. We have to show that the corresponding relations between Gödel numbers of formulas, sequences of Gödel numbers of formulas (which code sequences of formulas), and Gödel numbers of sequents, are primitive recursive.

1. $\Gamma \Rightarrow \Delta$ 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

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

2. Here we have to show that for each rule of inference R the relation $\mathrm{FollowsBy}_R(s, s')$ which holds if s and s' are the Gödel numbers of conclusion and premise of a correct application of R is primitive recursive. If R has two premises, $\mathrm{FollowsBy}_R$ of course has three arguments.

   For instance, $\Gamma \Rightarrow \Delta$ follows correctly from $\Gamma' \Rightarrow \Delta'$ by $\exists\mathrm{R}$ iff $\Gamma = \Gamma'$ and there is a sequence of formulas Δ'', a formula φ, a variable x and a closed term t such that $\Delta' = \Delta'', \varphi[t/x]$ and $\Delta = \Delta'', \exists x\, \varphi$. We just have to translate this into Gödel numbers. If $s = {}^{\#}\Gamma \Rightarrow \Delta^{\#}$ then $(s)_0 = {}^{\#}\Gamma^{\#}$ and $(s)_1 = {}^{\#}\Delta^{\#}$. So, $\mathrm{FollowsBy}_{\exists\mathrm{R}}(s, s')$ holds iff

$$\begin{aligned}&(s)_0 = (s')_0 \wedge \\ &(\exists d < s)\ (\exists f < s)\ (\exists x < s)\ (\exists t < s')\ (\mathrm{Frm}(f) \wedge \mathrm{Var}(y) \wedge \mathrm{Term}(t) \wedge \\ &\qquad (s')_1 = d \frown \langle \mathrm{Subst}(f, t, x) \rangle \wedge \\ &\qquad (s)_1 = d \frown \langle \#(\exists) \frown y \frown f \rangle\end{aligned}$$

   The individual lines express, respectively, "$\Gamma = \Gamma$," "there is a sequence (Δ'') with Gödel number d, a formula (φ) with Gödel number f, a variable with Gödel number x, and a term with Gödel number t," "$\Delta' = \Delta'', \varphi[t/x]$," and "$\Delta = \Delta'', \exists x\, \varphi$". (Remember that ${}^{\#}\Delta^{\#}$ is the number of a sequence of Gödel numbers of formulas in Δ.)

3. We first define a helper relation $\mathrm{hDeriv}(s, n)$ which holds if s codes a correct derivation to at least n inferences up from the end sequent. If $n = 0$ we let the relation be satisfied by default. Otherwise, $\mathrm{hDeriv}(s, n+1)$ iff either s consists just of an initial sequent, or it ends in a correct inference and the codes of the immediate subderivations satisfy $\mathrm{hDeriv}(s, n)$.

$$
\begin{aligned}
&\mathrm{hDeriv}(s, 0) \Leftrightarrow \mathrm{true} \\
&\mathrm{hDeriv}(s, n+1) \Leftrightarrow \\
&\qquad ((s)_0 = 0 \wedge \mathrm{InitialSeq}((s)_1)) \vee \\
&\qquad ((s)_0 = 1 \wedge \\
&\qquad\quad ((s)_2 = 1 \wedge \mathrm{FollowsBy}_{\mathrm{CL}}((s)_1, ((s)_3)_1)) \vee \\
&\qquad\quad \vdots \\
&\qquad\quad ((s)_2 = 16 \wedge \mathrm{FollowsBy}_{=}((s)_1, ((s)_3)_1)) \wedge \\
&\qquad\quad \mathrm{hDeriv}((s)_3, n)) \vee \\
&\qquad ((s)_0 = 2 \wedge \\
&\qquad\quad ((s)_2 = 1 \wedge \mathrm{FollowsBy}_{\mathrm{Cut}}((s)_1, ((s)_3)_1), ((s)_4)_1)) \vee \\
&\qquad\quad \vdots \\
&\qquad\quad ((s)_2 = 4 \wedge \mathrm{FollowsBy}_{\to\mathrm{L}}((s)_1, ((s)_3)_1), ((s)_4)_1)) \wedge \\
&\qquad\quad \mathrm{hDeriv}((s)_3, n) \wedge \mathrm{hDeriv}((s)_4, n))
\end{aligned}
$$

This is a primitive recursive definition. If the number n is large enough, e.g., larger than the maximum number of inferences between an initial sequent and the end sequent in s, it holds of s iff s is the Gödel number of a correct derivation. The number s itself is larger than that maximum number of inferences. So we can now define $\mathrm{Deriv}(s)$ by $\mathrm{hDeriv}(s, s)$.

□

Problem art.6. Define the following relations as in Proposition art.14:

1. $\mathrm{FollowsBy}_{\wedge\mathrm{R}}(s, s', s'')$,
2. $\mathrm{FollowsBy}_{=}(s, s')$,
3. $\mathrm{FollowsBy}_{\forall\mathrm{R}}(s, s')$.

Proposition art.15. *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 x 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 $(x)_1$ is the code of the end sequent of π, and so $((x)_1)_0$ is the code of the left side of the end sequent and $((x)_1)_1$ the right side. So we can express "the right side of the end sequent of π is φ" as $\mathrm{len}(((x)_1)_1) = 1 \wedge (((x)_1)_1)_0 = x$. 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

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

□

art.7 Derivations in Natural Deduction

inc:art:pnd:sec

In order to arithmetize derivations, we must represent derivations as numbers. 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 end-formula, the labels, and the representations of the sub-derivations leading to the premises of the last inference.

explanation

Definition art.16. If δ is a derivation in natural deduction, then ${}^{\ulcorner}\delta^{\#}$ is

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

2. $\langle 1, {}^{\ulcorner}\varphi^{\#}, n, k, {}^{\ulcorner}\delta_1{}^{\#}\rangle$ if δ ends in an inference with one premise, k is given by the following table according to which rule was used in the last inference, and δ_1 is the immediate subproof ending in the premise of the last inference. n is the label of the inference, or 0 if the inference does not discharge any assumptions.

Rule:	$\wedge$Elim	$\vee$Intro	$\to$Intro	$\neg$Intro	$\bot_I$
k:	1	2	3	4	5

Rule:	$\bot_C$	$\forall$Intro	$\forall$Elim	$\exists$Intro	$=$Intro
k:	6	7	8	9	10

3. $\langle 2, {}^{\ulcorner}\varphi^{\#}, n, k, {}^{\ulcorner}\delta_1{}^{\#}, {}^{\ulcorner}\delta_2{}^{\#}\rangle$ if δ ends in an inference with two premises, k is given by the following table according to which rule was used in the last inference, and δ_1, δ_2 are the immediate subderivations ending in the left and right premise of the last inference, respectively. n is the label of the inference, or 0 if the inference does not discharge any assumptions.

Rule:	$\wedge$Intro	$\to$Elim	$\neg$Elim
k:	1	2	3

4. $\langle 3, {}^{\#}\varphi^{\#}, n, {}^{\#}\delta_1{}^{\#}, {}^{\#}\delta_2{}^{\#}, {}^{\#}\delta_3{}^{\#}\rangle$ if δ ends in an $\vee$Elim inference. δ_1, δ_2, δ_3 are the immediate subderivations ending in the left, middle, and right premise of the last inference, respectively, and n is the label of the inference.

Example art.17. Consider the very simple derivation

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

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 $d_1 = \langle 1, {}^{\#}\varphi^{\#}, 0, 1, d_0\rangle$ (1 since $\wedge$Elim has one premise, Gödel number of conclusion φ, 0 because no assumption is discharged, 1 is the number coding $\wedge$Elim). The Gödel number of the entire derivation then is $\langle 1, {}^{\#}(\varphi \to \psi)^{\#}, 1, 3, d_1\rangle$, i.e.,

$$2^2 \cdot 3^{{}^{\#}(\varphi\to\psi)^{\#}+1} \cdot 5^2 \cdot 7^4 \cdot 11^{(2^2\cdot 3^{{}^{\#}\varphi^{\#}+1}\cdot 5^1\cdot 7^2\cdot 11^{(2^1\cdot 3^{{}^{\#}(\varphi\wedge\psi)^{\#}+1}\cdot 5^2)})}.$$

explanation

Having settled on a representation of derivations, we must also show that we can manipulate such derivations primite recursively, and express their essential properties and relations so. Some operations are simple: e.g., given a Gödel number d of a derivation, $(d)_1$ 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 on the Gödel numbers of δ and φ.

Proposition art.18. *The following relations are primitive recursive:*

inc:art:pnd: prop:followsby

1. *φ occurs as an assumption in δ with label n.*
2. *All assumption in δ with label n are of the form φ (i.e., we can discharge the assumption φ using label n in δ).*
3. *φ is an undischarged assumption of δ.*
4. *An inference with conclusion φ, upper derivations δ_1 (and δ_2, δ_3), and discharge label n is correct.*
5. *δ is a correct natural deduction derivation.*

Proof. We have to show that the corresponding relations between Gödel numbers of formulas, sequences of Gödel numbers of formulas (which code sets 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. For this we need a helper relation $\mathrm{hAssum}(x, d, n, i)$ which holds if the formula φ with Gödel number x occurs as an initial

formula with label n in the derivation with Gödel number d within i inferences up from the end-formula.

$$\begin{aligned}
&\mathrm{hAssum}(x, d, n, 0) \Leftrightarrow 1 \\
&\mathrm{hAssum}(x, d, n, i+1) \Leftrightarrow \\
&\qquad \mathrm{Sent}(x) \wedge (d = \langle 0, x, n\rangle \vee \\
&\qquad ((d)_0 = 1 \wedge \mathrm{hAssum}(x, (d)_4, n, i)) \vee \\
&\qquad ((d)_0 = 2 \wedge (\mathrm{hAssum}(x, (d)_4, n, i) \vee \\
&\qquad\qquad \mathrm{hAssum}(x, (d)_5, n, i))) \vee \\
&\qquad ((d)_0 = 3 \wedge (\mathrm{hAssum}(x, (d)_3, n, i) \vee \\
&\qquad\qquad \mathrm{hAssum}(x, (d)_2, n, i)) \vee \mathrm{hAssum}(x, (d)_3, n, i))
\end{aligned}$$

If the number i is large enough, e.g., larger than the maximum number of inferences between an initial formula and the end-formula of δ, it holds of x, d, n, and i iff φ is an initial formula in δ labelled n. The number d itself is larger than that maximum number of inferences. So we can define

$$\mathrm{Assum}(x, d, n) = \mathrm{hAssum}(x, d, n, d).$$

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 $(\forall y < d)\ (\mathrm{Assum}(y, d, n) \to y = x)$.

3. An occurrence of an assumption is not open if it occurs with label n in a subderivation that ends in a rule with discharge label n. Define the helper relation $\mathrm{hNotOpen}(x, d, n, i)$ as

$$\begin{aligned}
&\mathrm{hNotOpen}(x, d, n, 0) \Leftrightarrow 1 \\
&\mathrm{hNotOpen}(x, d, n, i+1) \Leftrightarrow \\
&\qquad (d)_2 = n \vee \\
&\qquad ((d)_0 = 1 \wedge \mathrm{hNotOpen}(x, (d)_4, n, i)) \vee \\
&\qquad ((d)_0 = 2 \wedge \mathrm{hNotOpen}(x, (d)_4, n, i) \wedge \\
&\qquad\qquad \mathrm{hNotOpen}(x, (d)_5, n, i))) \vee \\
&\qquad ((d)_0 = 3 \wedge \mathrm{hNotOpen}(x, (d)_3, n, i) \wedge \\
&\qquad\qquad \mathrm{hNotOpen}(x, (d)_4, n, i) \wedge \mathrm{hNotOpen}(x, (d)_5, n, i))
\end{aligned}$$

Note that all assumptions of the form φ labelled n are discharged in δ iff either the last inference of δ discharges them (i.e., the last inference has label n), or if it is discharged in all of the immediate subderivations.

A formula φ is an open assumption of δ iff it is an initial formula of δ (with label n) and is not discharged in δ (by a rule with lable n). We can then define $\mathrm{OpenAssum}(x, d)$ as

$$(\exists n < d)\ (\mathrm{Assum}(x, d, n, d) \wedge \neg \mathrm{hNotOpen}(x, d, n, d)).$$

4. Here we have to show that for each rule of inference R the relation $\mathrm{FollowsBy}_R(x, d_1, n)$ which holds if x is the Gödel number of the conclusion and d_1 is the Gödel number of a derivation ending in the premise of a correct application of R with label n is primitive recursive, and similarly for rules with two or three premises.

   The simplest case is that of the =Intro rule. Here there is no premise, i.e., $d_1 = 0$. However, φ must be of the form $t = t$, for a closed term t. Here, a primitive recursive definition is

$$(\exists t < x)\ (\mathrm{ClTerm}(t) \wedge x = ({}^{\ulcorner}{=}({}^{\#} \frown t \frown {}^{\urcorner}, {}^{\#} \frown t \frown {}^{\urcorner})^{\#})) \wedge d_1 = 0).$$

   For a more complicated example, $\mathrm{FollowsBy}_{\to\mathrm{Intro}}(x, d_1, n)$ holds iff φ is of the form $(\psi \to \chi)$, the end-formula of δ is χ, and any initial formula in δ labelled n is of the form ψ. We can express this primitive recursively by

$$\begin{aligned}(\exists y < x)\ (\mathrm{Sent}(y) \wedge \mathrm{Discharge}(y, d_1) \wedge \\ (\exists z < x)\ (\mathrm{Sent}(y) \wedge (d)_1 = z) \wedge \\ x = ({}^{\ulcorner}({}^{\#} \frown y \frown {}^{\urcorner}{\to}{}^{\#} \frown z \frown {}^{\urcorner})^{\#}))\end{aligned}$$

   (Think of y as the Gödel number of ψ and z as that of χ.)

   For another example, consider ∃Intro. Here, φ is the conclusion of a correct inference with one upper derivation iff there is a formula ψ, a closed term t and a variable x such that $\psi[t/x]$ is the end-formula of the upper derivation and $\exists x\, \psi$ is the conclusion φ, i.e., the formula with Gödel number x. So $\mathrm{FollowsBy}_{\exists\mathrm{Intro}}(x, d_1, n)$ holds iff

$$\begin{aligned}\mathrm{Sent}(x) \wedge (\exists y < x)\ (\exists v < x)\ (\exists t < d)\ (\mathrm{Frm}(y) \wedge \mathrm{Term}(t) \wedge \mathrm{Var}(v) \wedge \\ \mathrm{FreeFor}(y, t, v) \wedge \mathrm{Subst}(y, t, v) = (d_1)_1 \wedge x = ({}^{\ulcorner}\exists{}^{\#} \frown v \frown z))\end{aligned}$$

5. We first define a helper relation $\mathrm{hDeriv}(d, i)$ which holds if d codes a correct derivation at least to i inferences up from the end sequent. $\mathrm{hDeriv}(d, 0)$ holds always. Otherwise, $\mathrm{hDeriv}(d, i + 1)$ iff either d just codes an assumption or d ends in a correct inference and the codes of the immediate

sub-derivations satisfy $\mathrm{hDeriv}(d', i)$.

$$
\begin{aligned}
&\mathrm{hDeriv}(d, 0) \Leftrightarrow 1\\
&\mathrm{hDeriv}(d, i+1) \Leftrightarrow\\
&\qquad (\exists x < d)\ (\exists n < d)\ (\mathrm{Sent}(x) \wedge d = \langle 0, x, n\rangle) \vee\\
&\qquad ((d)_0 = 1 \wedge\\
&\qquad\quad ((d)_3 = 1 \wedge \mathrm{FollowsBy}_{\wedge\mathrm{Elim}}((d)_1, (d)_4, (d)_2) \vee\\
&\qquad\qquad \vdots\\
&\qquad\quad ((d)_3 = 10 \wedge \mathrm{FollowsBy}_{=\mathrm{Intro}}((d)_1, (d)_4, (d)_2)) \wedge\\
&\qquad\quad \mathrm{nDeriv}((d)_4, i)) \vee\\
&\qquad ((d)_0 = 2 \wedge\\
&\qquad\quad ((d)_3 = 1 \wedge \mathrm{FollowsBy}_{\wedge\mathrm{Intro}}((d)_1, (d)_4, (d)_5, (d)_2)) \vee\\
&\qquad\qquad \vdots\\
&\qquad\quad ((d)_3 = 3 \wedge \mathrm{FollowsBy}_{\neg\mathrm{Elim}}((d)_1, (d)_4, (d)_5, (d)_2)) \wedge\\
&\qquad\quad \mathrm{hDeriv}((d)_4, i) \wedge \mathrm{hDeriv}((d)_5, i)) \vee\\
&\qquad ((d)_0 = 3 \wedge\\
&\qquad\quad \mathrm{FollowsBy}_{\vee\mathrm{Elim}}((d)_1, (d)_3, (d)_4, (d)_5, (d)_2) \wedge\\
&\qquad\quad \mathrm{hDeriv}((d)_3, i) \wedge \mathrm{hDeriv}((d)_4, i)) \wedge \mathrm{hDeriv}((d)_5, i)
\end{aligned}
$$

This is a primitive recursive definition. Again we can define $\mathrm{Deriv}(d)$ as $\mathrm{hDeriv}(d, d)$.

□

Problem art.7. Define the following relations as in Proposition art.18:

1. $\mathrm{FollowsBy}_{\to\mathrm{Elim}}(x, d_1, d_2, n)$,
2. $\mathrm{FollowsBy}_{=\mathrm{Elim}}(x, d_1, d_2, n)$,
3. $\mathrm{FollowsBy}_{\vee\mathrm{Elim}}(x, d_1, d_2, d_3, n)$,
4. $\mathrm{FollowsBy}_{\forall\mathrm{Intro}}(x, d_1, n)$.

For the last one, you will have to also show that you can test primitive recursively if the formula with Gödel number x and the derivation with Gödel number d satisfy the eigenvariable condition, i.e., the eigenvariable a of the $\forall$Intro inference occurs neither in x nor in an open assumption of d.

Proposition art.19. *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 the previous proposition, the property $\mathrm{Deriv}(x)$ which holds iff x is the code of a correct derivation δ in natural deduction is primitive recursive. If x is such a code, then $(x)_1$ is the code of the end-formula of δ. Thus we can define $\mathrm{Prf}_\Gamma(x, y)$ by

$$\begin{aligned}\mathrm{Prf}_\Gamma(x, y) \Leftrightarrow \mathrm{Deriv}(x) \wedge (x)_1 = y \wedge \\ (\forall z < x)\ (\mathrm{OpenAssum}(z, x) \to R_\Gamma(z))\end{aligned}$$

□

Photo Credits

Bibliography