

Proving Gödel's Incompleteness Theorems

Lawrence C. Paulson

Computer Laboratory, University of Cambridge, England
lp15@cl.cam.ac.uk

1 Introduction

Gödel's first incompleteness theorem [3] states that every consistent theory containing a sufficient amount of mathematics has an undecidable sentence, one that although true can neither be proved nor disproved. His second incompleteness theorem states that in such a theory, the consistency of that theory cannot be proved.

The first theorem has been proved mechanically a number of times, originally in 1986 [8, 9]. The second theorem has never been proved mechanically until now. Another interesting aspect of these new proofs is the combined use of two treatments of variable syntax: nominal and de Bruijn. Some of the logical aspects of these proofs have been presented in a companion paper [7]. Details of the Isabelle formalisation are available in the full paper [6].

The machine proofs closely follow Świerczkowski [10]. Instead of the usual Peano arithmetic, he adopts the theory of hereditarily finite sets. An HF set is recursively defined to be a finite set of HF sets. The empty set is 0 and a nonempty set has the form $x \triangleleft y$, which denotes $x \cup \{y\}$. The advantage is that the coding of syntax can be done in a traditional set-theoretic fashion without having to resort to the Chinese remainder theorem or analogous techniques. Those would require the formalisation of a significant body of number theory: not the existing library versions of this material, but new proofs within a formalised calculus.

Nominal syntax is described at length elsewhere [11]. It deals with names and even sets of names, not necessarily those of bound variables, based on the idea of renaming permutations. In these proofs, definitions involved dozens of bound variables in a formal calculus; the nominal package coped reasonably well, but there were performance issues.

2 Gödel's First Incompleteness Theorem

Let us examine the proof step-by-step, interleaving the original informal arguments with elements of the Isabelle formalisation.

2.1 Formal Logical Syntax

The syntax of a first-order theory is formalised: terms, formulas, substitution, semantics (evaluation) and a deductive calculus. Here is the syntax of terms in the HF theory (0, variables, $\triangleleft$):

nominal_datatype *tm* = Zero | Var *name* | Eats *tm tm*

The HF formulas are $t \in u$, $t = u$, $\phi \vee \psi$, $\neg\phi$ or $\exists x [\phi]$:

nominal_datatype *fm* =
 Mem *tm tm* (infixr "IN" 150)
 | Eq *tm tm* (infixr "EQ" 150)
 | Disj *fm fm* (infixr "OR" 130)
 | Neg *fm*
 | Ex *x::name f::fm* binds *x* in *f*

This is a *nominal* datatype. Note that *x* denotes a variable bound by *Ex*.

Substituting the term *x* for the variable *i* in the formula *A* is written $A(i::=x)$. Thanks to the nominal package, it can be defined by structural recursion, even in the existential case, where we can straightforwardly state the necessary condition on the bound variable. Here, $\text{atom } j \# (i, x)$ essentially states that *j* must be fresh with respect to *i* and *x*.

nominal_primrec *subst_fm* :: "*fm* $\Rightarrow$ *name* $\Rightarrow$ *tm* $\Rightarrow$ *fm*"
where
 Mem: "(Mem *t u*)(*i*::=*x*) = Mem (subst *i x t*) (subst *i x u*)"
 | Eq: "(Eq *t u*)(*i*::=*x*) = Eq (subst *i x t*) (subst *i x u*)"
 | Disj: "(Disj *A B*)(*i*::=*x*) = Disj (*A*(*i*::=*x*)) (*B*(*i*::=*x*))"
 | Neg: "(Neg *A*)(*i*::=*x*) = Neg (*A*(*i*::=*x*))"
 | Ex: " $\text{atom } j \# (i, x) \implies (\text{Ex } j \text{ } A)(i::=x) = \text{Ex } j \text{ } (A(i::=x))$ "

Proofs about substitution are generally very simple, requiring nominal induction (which deals with the variable conditions) and simplification only.

2.2 A Sequent Calculus for HF Set Theory

Here is a Hilbert calculus, including axioms for disjunction and negation, specialisation axioms of the form $\phi(t/x) \rightarrow \exists x \phi$ and equality axioms.

inductive *hfthm* :: "*fm set* $\Rightarrow$ *fm* $\Rightarrow$ *bool*" (infixl "⊢" 55)
where
 Hyp: " $A \in H \implies H \vdash A$ "
 | Extra: " $H \vdash \text{extra_axiom}$ "
 | Bool: " $A \in \text{boolean_axioms} \implies H \vdash A$ "
 | Eq: " $A \in \text{equality_axioms} \implies H \vdash A$ "
 | Spec: " $A \in \text{special_axioms} \implies H \vdash A$ "
 | HF: " $A \in \text{HF_axioms} \implies H \vdash A$ "
 | Ind: " $A \in \text{induction_axioms} \implies H \vdash A$ "
 | MP: " $H \vdash A \text{ IMP } B \implies H' \vdash A \implies H \cup H' \vdash B$ "
 | Exists: " $H \vdash A \text{ IMP } B \implies \text{atom } i \# B \implies \forall C \in H. \text{atom } i \# C \implies H \vdash (\text{Ex } i \text{ } A) \text{ IMP } B$ "

Soundness is easily proved with respect to a traditional semantics, and the deduction theorem is also easily obtained.

theorem *deduction*: assumes "*insert A H* $\vdash$ *B*" shows "*H* $\vdash$ *A IMP B*"

From this, it is possible to derive a sequent calculus that is actually usable, and a number of results in HF set theory need to be proved in this calculus. These include basic notions such as the subset relation, unions, natural numbers and functions including a form of primitive recursion.

2.3 The Notion of Σ Formulas

Σ formulas [2, 10] are (inductively) all formulas of the form $t \in u$, $t = u$, $\alpha \vee \beta$, $\alpha \wedge \beta$, $\exists x \alpha$ and $(\forall x \in t) \alpha$; they include nearly all formulas used in the coding of syntax. Our proof of the second incompleteness theorem is by induction over Σ formulas. To simplify this proof, we require a restricted notion of *strict* Σ formulas, allowing only variables rather than general terms:

```

inductive ss_fm :: "fm  $\Rightarrow$  bool" where
  MemI:  "ss_fm (Var i IN Var j)"
| DisjI: "ss_fm A  $\implies$  ss_fm B  $\implies$  ss_fm (A OR B)"
| ConjI: "ss_fm A  $\implies$  ss_fm B  $\implies$  ss_fm (A AND B)"
| ExI:   "ss_fm A  $\implies$  ss_fm (Ex i A)"
| All2I: "ss_fm A  $\implies$  atom j  $\nmid$  (i,A)  $\implies$  ss_fm (All2 i (Var j) A)"

```

A Σ formula is by definition one that is provably equivalent (in HF) to some strict Σ formula containing no additional free variables.

```

definition Sigma_fm :: "fm  $\Rightarrow$  bool"
where "Sigma_fm A  $\longleftrightarrow$  ( $\exists B$ . ss_fm B & supp B  $\subseteq$  supp A &  $\{ \} \vdash A$  IFF B)"

```

By induction, every true Σ sentence has a formal proof. This meta-theoretic result allows us to reduce many formal proofs to evaluation.

```

theorem Sigma_fm_imp_thm: "[Sigma_fm  $\alpha$ ; ground_fm  $\alpha$ ; eval_fm e0  $\alpha$ ]  $\implies$   $\{ \} \vdash \alpha$ "

```

2.4 Coding Terms, Formulas, Abstraction and Substitution

To avoid having to formalise the theory of nominal Isabelle within the calculus, a de Bruijn version of terms and formulas is declared, along with a function to translate the nominal versions into their de Bruijn analogues. These are then encoded as terms of HF set theory according to the following simple system: $\ulcorner 0 \urcorner = 0$, $\ulcorner x_i \urcorner = i + 1$, $\ulcorner \in \urcorner = \langle 0, 0 \rangle$, $\ulcorner \triangleleft \urcorner = \langle 0, 0, 0 \rangle$, $\ulcorner = \urcorner = \langle 0, 0, 0, 0 \rangle$, $\ulcorner \vee \urcorner = \langle 0, 0, 0, 0, 0 \rangle$, $\ulcorner \neg \urcorner = \langle 0, 0, 0, 0, 0, 0 \rangle$, $\ulcorner \exists \urcorner = \langle 0, 0, 0, 0, 0, 0, 0 \rangle$. For the k th de Bruijn index we use $\langle \langle 0, 0, 0, 0, 0, 0, 0, 0 \rangle, k \rangle$.

The overloaded syntax $\ulcorner \dots \urcorner$ is set up to refer to the codes of terms and formulas. After a little effort, we obtain facts such as these:

```

lemma quot_Zero: "[Zero] = Zero"
lemma quot_Var: "[Var x] = SUCC (ORD_OF (nat_of_name x))"
lemma quot_Eats: "[Eats x y] = HPair (HTuple 1) (HPair [x] [y])"
lemma quot_Eq: "[x EQ y] = HPair (HTuple 2) (HPair ([x]) ([y]))"
lemma quot_Disj: "[A OR B] = HPair (HTuple 3) (HPair ([A]) ([B]))"
lemma quot_Ex: "[Ex i A] = HPair (HTuple 5) (quot_dbfm (trans_fm [i] A))"

```

Here *HPair* constructs a term denoting a pair and *HTuple n* constructs an $(n+2)$ -tuple of zeros. Compare these results with the ones listed informally above.

2.5 Syntactic Predicates

Next we must define logical predicates corresponding to terms, formulas, substitution, etc., which requires lengthy technical definitions [3, 10] and detailed syntactic proofs. With de Bruijn syntax, the notion of abstraction over a variable is also necessary.

Consider the definition of the notion of a constant term. Recursion is handled by explicitly formalising the stages of the construction as a sequence of successively larger terms. This technique is used in many similar situations.

```
nominal_primrec SeqConstP :: "tm  $\Rightarrow$  tm  $\Rightarrow$  tm  $\Rightarrow$  fm"
  where "[[atom l  $\#$  (s,k,sl,m,n,sm,sn); atom sl  $\#$  (s,m,n,sm,sn);
          atom m  $\#$  (s,n,sm,sn); atom n  $\#$  (s,sm,sn);
          atom sm  $\#$  (s,sn); atom sn  $\#$  (s)]]  $\implies$ 
  SeqConstP s k t =
    LstSeqP s k t AND
    All2 l (SUCC k) (Ex sl (HPair (Var l) (Var sl) IN s AND (Var sl EQ Zero
OR
    Ex m (Ex n (Ex sm (Ex sn (Var m IN Var l AND Var n IN Var l AND
    HPair (Var m) (Var sm) IN s AND HPair (Var n) (Var sn) IN
s AND
    Var sl EQ Q_Eats (Var sm) (Var sn)))))))]"
```

There are five bound variables: *l*, *sl*, *m*, *sm*, *n*, *sn*, each of which must be constrained to be distinct from one another. Here we see the quantifier syntax and set theoretic primitives such as *LstSeqP* (denoting a sequence), whose definitions are omitted here.

Dozens of other predicates must be defined, and while textbooks often seemed to take their correctness for granted, they are essentially code. They require substantial, tedious correctness proofs.

2.6 Predicates for the Coding of Proofs

A proof step in the HF calculus is an axiom, an axiom scheme or an inference rule. Axioms and propositional inference rules are easy to recognise. But inference rules involving quantifiers are tricky to code: syntactic conditions including abstraction and substitution need to be checked. Ultimately we define the provability predicate, *PfP*. The correctness of *PfP* follows directly from that of the predicates formalising the axioms and rules:

```
theorem proved_iff_proved_PfP: "{ $\}$   $\vdash$   $\alpha \longleftrightarrow$  { $\}$   $\vdash$  PfP [ $\alpha$ ]"
```

Thus the predicate *PfP* corresponds exactly to deduction in the HF calculus. We have formalised the deductive calculus within itself. Moreover, *PfP* [α] is a Σ formula.

2.7 The First Incompleteness Theorem

The first incompleteness theorem needs a function H such that $H(\ulcorner A \urcorner) = \ulcorner \ulcorner A \urcorner \urcorner$ for all A . But our formalism does not allow the introduction of new function symbols, so H has been formalised as the relation HRP . We can show this fact:

lemma prove_HRP: `fixes A::fm shows "{ } ⊢ HRP [A] [⌈A⌋]"`

Proving Gödel's diagonal lemma requires a function K to substitute the code of a formula into itself. This function, again, is realised as a relation.

lemma prove_KRP: `"{ } ⊢ KRP [Var i] [A] [A(i::=⌈A⌋)]"`

It is essential to prove that KRP behaves like a function. The substitution predicates, KRP and HRP can be proved to be single valued within the formal calculus. These single-step proofs are tedious but conceptually simple. We obtain a key lemma:

lemma KRP_subst_fm: `"{KRP [Var i] [β] (Var j)} ⊢ Var j EQ [β(i::=⌈β⌋)]"`

The second part of the diagonal lemma states that the free variables of the diagonal formula are those of the original formula except i .

lemma diagonal:
obtains δ **where** `"{ } ⊢ δ IFF α(i::=⌈δ⌋)" "supp δ = supp α - {atom i}"`

And so we reach the proof of the first incompleteness theorem (Fig. 1).

3 Gödel's Second Incompleteness Theorem

The second incompleteness theorem [1] follows from the *Hilbert-Bernays derivability conditions* [2, p. 15][4, p. 73].

$$\text{If } \vdash \alpha, \text{ then } \vdash \text{Pf}(\ulcorner \alpha \urcorner) \quad (1)$$

$$\text{If } \vdash \text{Pf}(\ulcorner \alpha \rightarrow \beta \urcorner) \text{ and } \vdash \text{Pf}(\ulcorner \alpha \urcorner), \text{ then } \vdash \text{Pf}(\ulcorner \beta \urcorner) \quad (2)$$

$$\text{If } \alpha \text{ is a strict } \Sigma \text{ sentence, then } \vdash \alpha \rightarrow \text{Pf}(\ulcorner \alpha \urcorner) \quad (3)$$

(Here Pf is the usual notation for the provability predicate.) Of these, only (3) is conceptually difficult. It is proved by induction over the construction of α as a Σ formula. This requires generalising the statement above to allow the formula α to contain free variables. This requires complicated and lengthy deductions in the calculus.

3.1 Pseudo-Coding

Condition (3) can be proved by induction on α if the assertion is generalised so that α can have free variables, say $x_1, \dots, x_n$:

$$\vdash \alpha(x_1, \dots, x_n) \rightarrow \text{Pf}(\ulcorner \alpha(\mathbf{x}_1, \dots, \mathbf{x}_n) \urcorner)$$

```

theorem Goedel_I:
  assumes " $\neg \{ \} \vdash \text{Fls}$ "
  obtains  $\delta$  where " $\{ \} \vdash \delta$  IFF  $\text{Neg (PfP } [\delta])$ " " $\neg \{ \} \vdash \delta$ " " $\neg \{ \} \vdash \text{Neg } \delta$ "
    " $\text{eval\_fm } e \ \delta$ " " $\text{ground\_fm } \delta$ "
proof -
  fix  $i::\text{name}$ 
  obtain  $\delta$  where " $\{ \} \vdash \delta$  IFF  $\text{Neg ((PfP (Var } i))(i::=[\delta]))$ "
    and suppd: " $\text{supp } \delta = \text{supp (Neg (PfP (Var } i)))} - \{ \text{atom } i \}$ "
    by (metis SyntaxN.Neg diagonal)
  hence diag: " $\{ \} \vdash \delta$  IFF  $\text{Neg (PfP } [\delta])$ "
    by simp
  hence np: " $\neg \{ \} \vdash \delta$ "
    by (metis assms Iff_MP_same Neg_D proved_iff_proved_PfP)
  hence npn: " $\neg \{ \} \vdash \text{Neg } \delta$ " using diag
    by (metis Iff_MP_same NegNeg_D Neg_cong proved_iff_proved_PfP)
  moreover have " $\text{eval\_fm } e \ \delta$ " using hfthm_sound [where  $e=e$ , OF diag]
    by simp (metis Pf_quot_imp_is_proved np)
  moreover have " $\text{ground\_fm } \delta$ " using suppd
    by (simp add: supp_conv_fresh ground_fm_aux_def subset.eq)
    (metis fresh_ineq_at_base)
  ultimately show ?thesis
    by (metis diag np npn that)
qed

```

Fig. 1. Gödel's First Incompleteness Theorem

The syntactic constructions used in this formula have to be formalised, and the necessary transformations have to be justified within the HF calculus. Here we write $\mathbf{x}$ to denote some term that codes the value of x . Note that $\ulcorner \alpha(\mathbf{x}_1, \dots, \mathbf{x}_n) \urcorner$ is a function of $x_1, \dots, x_n$.

First, we must generalise coding to allow certain variables to be preserved as real variables in the coded term, not replaced by coded variable names. The V -code of a formula α , where V is a set of variables to be preserved, is written $\lfloor \alpha \rfloor V$. V -coding is otherwise identical to standard coding.

Formalising the notation $\ulcorner \alpha(\mathbf{x}_1, \dots, \mathbf{x}_n) \urcorner$ requires a function for simultaneous substitution. The boldface symbols are formalised as new variables. The nominal primitives manage the mapping from old to new variables. The substitution is well-defined only because the boldface symbols denote terms representing the codes of constants: not themselves necessarily ground terms, but denoting the codes of ground terms. Ultimately, we can prove that, if β is a theorem in HF (if $\vdash \text{Pf } \beta$ holds) then the result of substituting constants for its variables is also an HF theorem.

A major chore is proving that the various syntactic predicates can be executed. We need to reason about sequences and their truncation and concatenation in order to show that the treatment of recursion actually works.

3.2 Quoted Values

Formalising expressions like $x = y \rightarrow \text{Pf } \ulcorner \mathbf{x} = \mathbf{y} \urcorner$ requires a function Q such that $Q(x) = \ulcorner x \urcorner$:

$$\begin{aligned} Q(0) &= \ulcorner 0 \urcorner = 0 \\ Q(x \triangleleft y) &= \langle \ulcorner \triangleleft \urcorner, Q(x), Q(y) \rangle \end{aligned}$$

This recursion is ambiguous without a total ordering on sets, which is difficult to define within the HF calculus. But we can make do with the (non-single-valued) relation between a set x and (the codes of) terms that denote x .

The relation *QuoteP* can be defined using precisely the same methods as before. We can prove the following facts.

lemma *QuoteP_Zero*: " $\{\} \vdash \text{QuoteP Zero Zero}$ "

lemma *QuoteP_Eats*:

" $\{\text{QuoteP } t1 \ u1, \text{QuoteP } t2 \ u2\} \vdash \text{QuoteP (Eats } t1 \ t2) (Q_Eats \ u1 \ u2)$ "

lemma *QuoteP_imp_ConstP*: " $\{\text{QuoteP } x \ y\} \vdash \text{ConstP } y$ "

This last lemma states that any result given by Q is the code of a constant, and is necessary to use this primitive with the pseudo-coding results of the previous section.

Let $\vdash \beta$ be a theorem of HF whose free variables belong to the set Vs . Take the code of this theorem, $\lfloor \beta \rfloor$, and replace some subset $V \subseteq Vs$ of its free variables by new variables constrained by the *QuoteP* relation. Then the result of this substitution is also the code of a theorem: it satisfies the provability predicate. The formal statement of this result, which requires additional notation, can be found in the full paper.

3.3 Proving $\vdash \alpha \rightarrow \text{Pf}(\ulcorner \alpha \urcorner)$

Condition (3) is proved by induction on the structure of α . As stated in Sect. 2.3 above, a strict Σ formula has the form $x \in y$, $\alpha \vee \beta$, $\alpha \wedge \beta$, $\exists x \alpha$ or $(\forall x \in y) \alpha$. Therefore, the induction will include one single base case,

$$x \in y \rightarrow \text{Pf} \ulcorner x \in y \urcorner,$$

along with inductive steps for disjunction, conjunction, etc.

A key step is the following lemma:

$$x = y \rightarrow \text{Pf} \ulcorner x = y \urcorner$$

It is tricky to prove because our Q is not a true function. However, the conjunction

$$(x \in y \rightarrow \text{Pf} \ulcorner x \in y \urcorner) \wedge (x \subseteq y \rightarrow \text{Pf} \ulcorner x \subseteq y \urcorner)$$

can be proved by induction (in HF) on the sum of the sizes of x and y . The required lemma follows easily. These issues arise only with HF, which lacks an easily-definable ordering; the approach sketched above, dispensing with the need for Q to be single-valued, appears to be unique in the literature on Gödel's theorem.

The main induction on α requires many delicate steps as new variables (with their partners) are introduced in the quantifier cases. The inductive lemma is complicated by technical conditions involving variable names and permutations. But as a special case, we obtain condition (3).

theorem *Provability*:

assumes *"Sigma_fm α " "ground_fm α "*
shows *" $\{\alpha\} \vdash \text{PfP} \ulcorner \alpha \urcorner$ "*

Here $\alpha \vdash \text{Pf} \ulcorner \alpha \urcorner$ is equivalent to $\vdash \alpha \rightarrow \text{Pf} \ulcorner \alpha \urcorner$. Note that this theorem is much stronger than the previously proved $\vdash \alpha \implies \vdash \text{Pf} \ulcorner \alpha \urcorner$, which holds by induction on the proof of $\vdash \alpha$.

3.4 The Second Incompleteness Theorem

The second incompleteness theorem now follows (Fig. 2). The diagonal formula, δ , is obtained via the first incompleteness theorem. Then $\text{Pf} \ulcorner \delta \urcorner \vdash \text{Pf}(\ulcorner \text{Pf} \ulcorner \delta \urcorner \urcorner)$ and $\text{Pf} \ulcorner \delta \urcorner \vdash \text{Pf}(\ulcorner \neg \text{Pf} \ulcorner \delta \urcorner \urcorner)$ hold, and therefore $\text{Pf} \ulcorner \delta \urcorner \vdash \text{Pf}(\ulcorner \perp \urcorner)$. It follows that if the system proves its own consistency, then it also proves $\vdash \neg \text{Pf} \ulcorner \delta \urcorner$ and therefore $\vdash \delta$, contradicting the first theorem.

Both of Gödel's incompleteness theorems have been formalised by machine for the first time. The proofs are available online [5]. Completing these proofs with a reasonable amount of effort required the use of a combination of techniques not seen before in the extensive literature on this subject.

```

theorem Goedel_II:
  assumes " $\neg \{ \} \vdash Fls$ "
  shows " $\neg \{ \} \vdash Neg (PfP [Fls])$ "
proof -
  from assms Goedel_I obtain  $\delta$ 
  where diag: " $\{ \} \vdash \delta$  IFF  $Neg (PfP [\delta])$ " " $\neg \{ \} \vdash \delta$ "
    and gnd: "ground_fm  $\delta$ "
    by metis
  have " $\{PfP [\delta]\} \vdash PfP [PfP [\delta]]$ "
    by (auto simp: Provability ground_fm_aux_def supp_conv_fresh)
  moreover have " $\{PfP [\delta]\} \vdash PfP [Neg (PfP [\delta])]$ "
    apply (rule MonPon_PfP_implies_PfP [OF _ gnd])
    apply (metis Conj_E2 Iff_def Iff_sym diag(1))
    apply (auto simp: ground_fm_aux_def supp_conv_fresh)
    done
  moreover have "ground_fm  $(PfP [\delta])$ "
    by (auto simp: ground_fm_aux_def supp_conv_fresh)
  ultimately have " $\{PfP [\delta]\} \vdash PfP [Fls]$ " using PfP_quot_contra
    by (metis (no_types) anti_deduction cut2)
  thus " $\neg \{ \} \vdash Neg (PfP [Fls])$ "
    by (metis Iff_MP2_same Neg_mono cut1 diag)
qed

```

Fig. 2. Gödel's Second Incompleteness Theorem

References

1. Joan Bagaria. A short guide to Gödel's second incompleteness theorem. *Teorema*, 22(3):5–15, 2003.
2. George Stephen Boolos. *The Logic of Provability*. Cambridge University Press, 1993.
3. Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.
4. Richard E Grandy. *Advanced Logic for Applications*. Reidel, 1977.
5. Lawrence C. Paulson. Gödel's incompleteness theorems. *Archive of Formal Proofs*, November 2013. <http://afp.sf.net/entries/Incompleteness.shtml>, Formal proof development.
6. Lawrence C. Paulson. A mechanised proof of Gödel's incompleteness theorems using Nominal Isabelle. Submitted for publication, 2013.
7. Lawrence C. Paulson. A machine-assisted proof of Gödel's incompleteness theorems for the theory of hereditarily finite sets. *Review of Symbolic Logic*, 2014. in press; available online.
8. Natarajan Shankar. *Proof-checking Metamathematics*. PhD thesis, University of Texas at Austin, 1986.
9. Natarajan Shankar. *Metamathematics, Machines, and Gödel's Proof*. Cambridge University Press, 1994.

10. S. Świerczkowski. Finite sets and Gödel's incompleteness theorems.
Dissertationes Mathematicae, 422:1–58, 2003.
<http://journals.impan.gov.pl/dm/Inf/422-0-1.html>.
11. Christian Urban and Cezary Kaliszyk. General bindings and alpha-equivalence in Nominal Isabelle. *Logical Methods in Computer Science*, 8(2:14):1–35, 2012.