From Abstract to Concrete Gödel's Incompleteness Theorems—Part I

Andrei Popescu Dmitriy Traytel

March 17, 2025

Abstract

We validate an abstract formulation of Gödel's First and Second Incompleteness Theorems from a separate AFP entry by instantiating them to the case of *finite sound extensions of the Hereditarily Finite (HF) Set theory*, i.e., FOL theories extending the HF Set theory with a finite set of axioms that are sound in the standard model. The concrete results had been previously formalised in an AFP entry by Larry Paulson; our instantiation reuses the infrastructure developed in that entry.

Contents

1 The Instantiation

definition *Fvars t* = {*a* :: *name*. ¬ *atom a* ♯ *t*}

lemma *Fvars_tm_simps*[*simp*]:
  Fvars Zero = {}
  Fvars (*Var a*) = {*a*}
  Fvars (*Eats x y*) = *Fvars x* ∪ *Fvars y*
  by (*auto simp*: *Fvars_def fresh_at_base*(*2*))

lemma *finite_Fvars_tm*[*simp*]:
  fixes *t* :: *tm*
  shows *finite* (*Fvars t*)
  by (*induct t rule*: *tm.induct*) *auto*

lemma *Fvars_fm_simps*[*simp*]:
  Fvars (*x IN y*) = *Fvars x* ∪ *Fvars y*
  Fvars (*x EQ y*) = *Fvars x* ∪ *Fvars y*
  Fvars (*A OR B*) = *Fvars A* ∪ *Fvars B*
  Fvars (*A AND B*) = *Fvars A* ∪ *Fvars B*
  Fvars (*A IMP B*) = *Fvars A* ∪ *Fvars B*
  Fvars Fls = {}
  Fvars (*Neg A*) = *Fvars A*
  Fvars (*Ex a A*) = *Fvars A* − {*a*}
  Fvars (*All a A*) = *Fvars A* − {*a*}
  by (*auto simp*: *Fvars_def fresh_at_base*(*2*))

lemma *finite_Fvars_fm*[*simp*]:
  fixes *A* :: *fm*
  shows *finite* (*Fvars A*)

```
  by (induct A rule: fm.induct) auto

lemma subst_tm_subst_tm[simp]:
  x ≠ y ⟹ atom x ♯ u ⟹ subst y u (subst x t v) = subst x (subst y u t) (subst y u v)
  by (induct v rule: tm.induct) auto

lemma subst_fm_subst_fm[simp]:
  x ≠ y ⟹ atom x ♯ u ⟹ (A(x::=t))(y::=u) = (A(y::=u))(x::=subst y u t)
  by (nominal_induct A avoiding: x t y u rule: fm.strong_induct) auto

lemma Fvars_ground_aux: Fvars t ⊆ B ⟹ ground_aux t (atom ‘ B)
  by (induct t rule: tm.induct) auto

lemma ground_Fvars: ground t ⟷ Fvars t = {}
  apply (rule iffI)
   apply (auto simp only: Fvars_def ground_fresh) []
  apply (auto intro: Fvars_ground_aux[of t {}, simplified])
  done

lemma Fvars_ground_fm_aux: Fvars A ⊆ B ⟹ ground_fm_aux A (atom ‘ B)
  apply (induct A arbitrary: B rule: fm.induct)
      apply (auto simp: Diff_subset_conv Fvars_ground_aux)
  apply (drule meta_spec, drule meta_mp, assumption)
  apply auto
  done

lemma ground_fm_Fvars: ground_fm A ⟷ Fvars A = {}
  apply (rule iffI)
   apply (auto simp only: Fvars_def ground_fresh) []
  apply (auto intro: Fvars_ground_fm_aux[of A {}, simplified])
  done

interpretation Generic_Syntax where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  apply unfold_locales
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal for t by (induct t rule: tm.induct) auto
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal unfolding Fvars_def fresh_subst_fm_if by auto
  subgoal unfolding Fvars_def by auto
  subgoal unfolding Fvars_def by simp
  subgoal by simp
```

subgoal unfolding *Fvars_def* **by** *simp*
done

lemma *coding_tm_Fvars_empty*[*simp*]: *coding_tm t* ⟹ *Fvars t* = {}
by (*induct t rule*: *coding_tm.induct*) (*auto simp*: *Fvars_def*)

lemma *Fvars_empty_ground*[*simp*]: *Fvars t* = {} ⟹ *ground t*
by (*induct t rule*: *tm.induct*) *auto*

interpretation *Syntax_with_Numerals* **where**
var = *UNIV* :: *name set*
and *trm* = *UNIV* :: *tm set*
and *fmla* = *UNIV* :: *fm set*
and *num* = {*t*. *ground t*}
and *Var* = *Var*
and *FvarsT* = *Fvars*
and *substT* = λ*t u x*. *subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x*. *subst_fm A x u*
apply *unfold_locales*
subgoal by (*auto intro*!: *exI*[*of _ Zero*])
subgoal by *simp*
subgoal by (*simp add*: *ground_Fvars*)
done

declare *FvarsT_num*[*simp del*]

interpretation *Deduct2_with_False* **where**
var = *UNIV* :: *name set*
and *trm* = *UNIV* :: *tm set*
and *fmla* = *UNIV* :: *fm set*
and *num* = {*t*. *ground t*}
and *Var* = *Var*
and *FvarsT* = *Fvars*
and *substT* = λ*t u x*. *subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x*. *subst_fm A x u*
and *eql* = (*EQ*)
and *cnj* = (*AND*)
and *imp* = (*IMP*)
and *all* = *All*
and *exi* = *Ex*
and *fls* = *Fls*
and *prv* = (⊢) {}
and *bprv* = (⊢) {}
apply *unfold_locales*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*
subgoal by *simp*

```
  subgoal unfolding Fvars_def by simp
  subgoal unfolding Fvars_def by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal using MP_null by blast
  subgoal by blast
  subgoal for A B C
    apply (rule Imp_I)+
    apply (rule MP_same[of _ B])
     apply (rule MP_same[of _ C])
      apply (auto intro: Neg_D)
    done
  subgoal by blast
  subgoal by blast
  subgoal by blast
  subgoal unfolding Fvars_def by (auto intro: MP_null)
  subgoal unfolding Fvars_def by (auto intro: MP_null)
  subgoal by (auto intro: All_D)
  subgoal by (auto intro: Ex_I)
  subgoal by simp
  subgoal by (metis Conj_E2 Iff_def Imp_I Var_Eq_subst_Iff)
  subgoal by blast
  subgoal by simp
  done

interpretation HBL1 where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and imp = (IMP)
  and all = All
  and exi = Ex
  and prv = (⊢) {}
  and bprv = (⊢) {}
  and enc = quot
  and P = PfP (Var xx)
  apply unfold_locales
  subgoal by (simp add: quot_fm_coding)
  subgoal by simp
  subgoal unfolding Fvars_def by (auto simp: fresh_at_base(2))
  subgoal by (auto simp: proved_imp_proved_PfP)
  done

interpretation Goedel_Form where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
```

```
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and imp = (IMP)
  and all = All
  and exi = Ex
  and fls = Fls
  and prv = (⊢) {}
  and bprv = (⊢) {}
  and enc = quot
  and S = KRP (quot (Var xx)) (Var xx) (Var yy)
  and P = PfP (Var xx)
  apply unfold_locales
  subgoal by simp
  subgoal unfolding Fvars_def by (auto simp: fresh_at_base(2))
  subgoal
    unfolding Let_def
    by (subst psubst_eq_rawpsubst2)
      (auto simp: quot_fm_coding prove_KRP Fvars_def)
  subgoal
    unfolding Let_def
    by (subst (1 2) psubst_eq_rawpsubst2)
      (auto simp: quot_fm_coding KRP_unique[THEN Sym] Fvars_def)
  done

interpretation g2: Goedel_Second_Assumptions where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and imp = (IMP)
  and all = All
  and exi = Ex
  and fls = Fls
  and prv = (⊢) {}
  and bprv = (⊢) {}
  and enc = quot
  and S = KRP (quot (Var xx)) (Var xx) (Var yy)
  and P = PfP (Var xx)
  apply unfold_locales
  subgoal by (auto simp: PP_def intro: PfP_implies_ModPon_PfP_quot)
  subgoal by (auto simp: PP_def quot_fm_coding Provability)
  done

theorem Goedel_II: ¬ {} ⊢ Fls ⟹ ¬ {} ⊢ neg (PfP «Fls»)
  by (rule g2.goedel_second[unfolded consistent_def PP_def PfP_subst subst.simps simp_thms if_True])
```

```
lemma ground_fm_PrfP[simp]:
  ground_fm (PrfP s k t) ⟷ ground s ∧ ground k ∧ ground t
  by (auto simp add: ground_aux_def ground_fm_aux_def supp_conv_fresh)


lemma Fvars_HPair[simp]: Fvars (HPair t u) = Fvars t ∪ Fvars u
  unfolding Fvars_def
  by auto

lemma ground_HPair[simp]: ground (HPair t u) ⟷ ground t ∧ ground u
  unfolding ground_Fvars
  by auto

interpretation dwfd: Deduct2_with_False_Disj where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and dsj = (OR)
  and imp = (IMP)
  and all = All
  and exi = Ex
  and fls = Fls
  and prv = (⊢) {}
  and bprv = (⊢) {}
  apply unfold_locales
  subgoal by simp
  subgoal by simp
  subgoal by simp
  subgoal by (auto intro: Disj_I1)
  subgoal by (auto intro: Disj_I2)
  subgoal by (auto intro: ContraAssume)
  subgoal by simp
  done

interpretation Minimal_Truth_Soundness where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and dsj = (OR)
  and imp = (IMP)
  and all = All
  and exi = Ex
```

and *fls* = *Fls*
and *prv* = (⊢) {}
and *isTrue* = *eval_fm e0*
apply *unfold_locales*
subgoal by (*auto simp*: *Fls_def*)
subgoal by *simp*
subgoal by (*auto simp only*: *ex_eval_fm_iff_exists_tm eval_fm.simps*(*4*) *subst_fm.simps*)
subgoal by (*auto simp only*: *ex_eval_fm_iff_exists_tm*)
subgoal by (*simp add*: *neg_def*)
subgoal by (*auto dest*: *hfthm_sound*)
done

lemma *neg_Neg*:
{} ⊢ *neg* φ *IFF Neg* φ
unfolding *neg_def*
by (*auto simp*: *Fls_def intro*: *ContraAssume*)

lemma *ground_aux_mono*: *A* ⊆ *B* ⟹ *ground_aux t A* ⟹ *ground_aux t B*
unfolding *ground_aux_def* by *auto*

interpretation *g1*: *Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical* where
var = *UNIV* :: *name set*
and *trm* = *UNIV* :: *tm set*
and *fmla* = *UNIV* :: *fm set*
and *num* = {*t*. *ground t*}
and *Var* = *Var*
and *FvarsT* = *Fvars*
and *substT* = λ*t u x*. *subst x u t*
and *Fvars* = *Fvars*
and *subst* = λ*A u x*. *subst_fm A x u*
and *eql* = (*EQ*)
and *cnj* = (*AND*)
and *dsj* = (*OR*)
and *imp* = (*IMP*)
and *all* = *All*
and *exi* = *Ex*
and *fls* = *Fls*
and *prv* = (⊢) {}
and *bprv* = (⊢) {}
and *enc* = *quot*
and *S* = *KRP* (*quot* (*Var xx*)) (*Var xx*) (*Var yy*)
and *P* = *PfP* (*Var xx*)
and *isTrue* = *eval_fm e0*
and *Pf* = *Ex xx′* (*Ex yy′* (*Var yy EQ HPair* (*Var xx′*) (*Var yy′*) *AND PrfP* (*Var xx′*) (*Var yy′*) (*Var xx*)))
apply *unfold_locales*
subgoal by *simp*
subgoal unfolding *Fvars_def* by (*auto simp*: *fresh_at_base*(*2*))
subgoal for φ
unfolding *Let_def*
supply *PfP.simps*[*simp del*]
apply (*subst psubst_eq_rawpsubst2*) apply (*simp_all add*: *PfP.simps*[*of yy′ xx′ quot* φ, *simplified*])
apply (*auto simp*: *eqv_def*)
apply (*rule Ex_I*[*of _ _ _ HPair* (*Var xx′*) (*Var yy′*)])
apply (*subst subst_fm_Ex_with_renaming*[*of xx _ xx′ yy*]; (*auto simp*: *Conj_eqvt*))
apply (*subst subst_fm_Ex_with_renaming*[*of zz _ yy′ yy*]; (*auto simp*: *Conj_eqvt HPair_eqvt PrfP.eqvt*))

```
    apply (rule Ex_I[of _ _ _ (Var xx')]; auto)
    apply (rule Ex_I[of _ _ _ (Var yy')]; auto)
    apply (rule Ex_I[of _ _ _ (Var yy')]; auto)
   apply (rule Ex_I[of _ _ _ (Var xx')]; auto)
   done
  subgoal
   by (auto simp: PP_def proved_iff_proved_PfP[symmetric])
  subgoal for n φ
   unfolding Let_def
   apply (subst (1 2) psubst_eq_rawpsubst2)
        apply (simp_all add: ground_Fvars)
   apply (rule impI)
   apply (rule Sigma_fm_imp_thm)
     apply (auto simp: ground_Fvars[symmetric] elim: ground_aux_mono[OF empty_subsetI])
    apply (auto simp: ground_aux_def ground_fm_aux_def supp_conv_fresh fresh_at_base Fvars_def)
   done
  subgoal for φ
   apply (rule NegNeg_D)
  apply (auto simp: PP_def dest!: Iff_MP_same[OF neg_Neg] Iff_MP_same[OF Neg_cong[OF neg_Neg]])
   done
  done

theorem Goedel_I: ∃φ. ¬ {} ⊢ φ ∧ ¬ {} ⊢ Neg φ ∧ eval_fm e0 φ
  by (meson Iff_MP2_same g1.recover_proofs.goedel_first_classic_strong[OF consistent] neg_Neg)
```

The following interpretation is redundant, because *Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_* (interpreted above) is a sublocale of *Goedel_Form_Classical_HBL1_rev_prv_Minimal_Truth_Soundness_TIP*. However, the latter requires less infrastructure (no Pf formula).

The definition of isTrue prevents Isabelle from noticing that the locale has already been interpreted via the above g1 interpretation of *Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical*.

```
definition isTrue where
  isTrue = eval_fm e0

interpretation g1': Goedel_Form_Classical_HBL1_rev_prv_Minimal_Truth_Soundness_TIP where
      var = UNIV :: name set
  and trm = UNIV :: tm set
  and fmla = UNIV :: fm set
  and num = {t. ground t}
  and Var = Var
  and FvarsT = Fvars
  and substT = λt u x. subst x u t
  and Fvars = Fvars
  and subst = λA u x. subst_fm A x u
  and eql = (EQ)
  and cnj = (AND)
  and dsj = (OR)
  and imp = (IMP)
  and all = All
  and exi = Ex
  and fls = Fls
  and prv = (⊢) {}
  and bprv = (⊢) {}
  and enc = quot
  and S = KRP (quot (Var xx)) (Var xx) (Var yy)
  and P = PfP (Var xx)
  and isTrue = isTrue
  apply unfold_locales
  unfolding isTrue_def
```

```
  subgoal by (auto simp: Fls_def)
  subgoal by simp
  subgoal by (auto simp only: ex_eval_fm_iff_exists_tm eval_fm.simps(4) subst_fm.simps)
  subgoal by (auto simp only: ex_eval_fm_iff_exists_tm)
  subgoal by (simp add: neg_def)
  subgoal by (auto dest: hfthm_sound)
  subgoal by (auto simp: proved_iff_proved_PfP[symmetric] PP_def quot_fm_coding
      simp del: eval_fm_PfP
      dest!: Iff_MP_same[OF neg_Neg] Iff_MP_same[OF Neg_cong[OF neg_Neg]] NegNeg_D
      Sigma_fm_imp_thm[rotated 2])
  done

theorem Goedel_I': ∃φ. ¬ {} ⊢ φ ∧ ¬ {} ⊢ Neg φ ∧ isTrue φ
  by (meson Iff_MP2_same g1'.goedel_first_classic_strong[OF consistent] neg_Neg)
```