

An Abstract Formalization of Gödel's Incompleteness Theorems

Andrei Popescu Dmitriy Traytel

September 13, 2023

Abstract

We present an abstract formalization of Gödel's incompleteness theorems. We analyze sufficient conditions for the theorems' applicability to a partially specified logic. Our abstract perspective enables a comparison between alternative approaches from the literature. These include Rosser's variation of the first theorem, Jeroslow's variation of the second theorem, and the Swierczkowski–Paulson semantics-based approach. This AFP entry is the main entry point to the results described in our CADE-27 paper [1].

As part of our abstract formalization's validation, we instantiate our locales twice in the separate AFP entries [Goedel_HFSet_Semantic](#) and [Goedel_HFSet_Semanticless](#).

Contents

1	Deduction with Two Provability Relations	3
1.1	From Deduction with One Provability Relation to Two	3
1.2	Factoring In Explicit Proofs	6
2	Abstract Encoding	8
3	Representability Assumptions	10
3.1	Representability of Negation	10
3.2	Representability of Self-Substitution	12
3.3	Representability of Self-Soft-Substitution	13
3.4	Clean Representability of the "Proof-of" Relation	15
4	Diagonalization	18
4.1	Alternative Diagonalization via Self-Substitution	18
4.2	Alternative Diagonalization via Soft Self-Substitution	18
5	The Hilbert-Bernays-Löb (HBL) Derivability Conditions	20
5.1	First Derivability Condition	20
5.2	Connections between Proof Representability, First Derivability Condition, and Its Converse	21
5.2.1	HBL1 from "proof-of" representability	21
5.2.2	Sufficient condition for the converse of HBL1	22
5.3	Second and Third Derivability Conditions	22
6	Gödel Formulas	24
7	Standard Models with Two Provability Relations	27
7.1	Proof recovery from <i>HBL1_iff</i>	28
8	Abstract Formulations of Gödel's First Incompleteness Theorem	34
8.1	Proof-Theoretic Versions of Gödel's First	34
8.1.1	The easy half	34
8.1.2	The hard half	34
8.2	Model-Theoretic Versions of Gödel's First	35
8.2.1	First model-theoretic version	35
8.2.2	Second model-theoretic version	36
8.3	Classical-Logic Versions of Gödel's First	37
8.3.1	First classical-logic version	37
8.3.2	Second classical-logic version	38
8.3.3	Third classical-logic version	39
9	Rosser Formulas	41

10 Abstract Formulations of Gödel-Rosser's First Incompleteness Theorem	45
10.1 Proof-Theoretic Versions	45
10.2 Model-Theoretic Versions	46
10.2.1 First model-theoretic version	46
10.2.2 Second model-theoretic version	47
11 Abstract Formulation of Gödel's Second Incompleteness Theorem	50
12 Jeroslow's Variant of Gödel's Second Incompleteness Theorem	52
12.1 Encodings and Derivability	52
12.1.1 Encoding of formulas	52
12.1.2 Encoding of computable functions	52
12.1.3 Term-encoding of computable functions	53
12.1.4 The first Hilbert-Bernays-Löb derivability condition	54
12.2 A Formalization of Jeroslow's Original Argument	54
12.2.1 Preliminaries	54
12.2.2 Jeroslow-style diagonalization	55
12.2.3 Jeroslow's Second Incompleteness Theorem	56
12.3 A Simplification of Jeroslow's Original Argument	58
12.3.1 Jeroslow-style term-based diagonalization	58
12.3.2 Term-based version of Jeroslow's Second Incompleteness Theorem	59
12.3.3 A variant of the Second Incompleteness Theorem	60
13 Löb Formulas	62
14 Löb's Theorem	64
15 Abstract Formulation of Tarski's Theorems	66
15.1 Non-Definability of Truth	66
15.2 Non-Expressiveness of Truth	67

Chapter 1

Deduction with Two Provability Relations

We work with two provability relations: provability *prv* and basic provability *bprv*.

1.1 From Deduction with One Provability Relation to Two

```
locale Deduct2 =
  Deduct
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv
+
  B: Deduct
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  bprv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and eql cnj imp all exi
and prv bprv
+
assumes bprv_prv:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{bprv } \varphi \implies \text{prv } \varphi$ 
begin

lemma bprv_prv':
  assumes  $\varphi: \varphi \in \text{fmla}$  and  $b: \text{bprv } \varphi$ 
  shows  $\text{prv } \varphi$ 
  <proof>

end — context Deduct2

locale Deduct2_with_False =
  Deduct_with_False
  var trm fmla Var FvarsT substT Fvars subst
```

```

    eql cnj imp all exi
    fls
    num
    prv
+
B: Deduct_with_False
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  num
  bprv
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and num
and prv bprv
+
assumes bprv_prv:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{bprv } \varphi \implies \text{prv } \varphi$ 

sublocale Deduct2_with_False < d_dwf: Deduct2
  <proof>

context Deduct2_with_False begin

lemma consistent_B_consistent: consistent  $\implies$  B.consistent
  <proof>

end — context Deduct2_with_False

locale Deduct2_with_False_Disj =
Deduct_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
+
B: Deduct_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  bprv
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
+
assumes bprv_prv:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{bprv } \varphi \implies \text{prv } \varphi$ 

```

sublocale *Deduct2_with_False_Disj* < *dwf_dwfd*: *Deduct2_with_False*
 <proof>

locale *Deduct2_with_PseudoOrder* =
Deduct2_with_False_Disj
 var *trm fmla Var FvarsT substT Fvars subst*
eql cnj imp all exi
fls
dsj
num
prv bprv
 +
Syntax_PseudoOrder
 var *trm fmla Var FvarsT substT Fvars subst*
eql cnj imp all exi
fls
dsj
num
Lq

for
 var :: 'var set **and** trm :: 'trm set **and** fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and Lq
 +
assumes

Lq_num:
 let *LLq* = (λ *t1 t2. psubst Lq [(t1,zz), (t2,yy)]) in
 $\forall \varphi \in \text{fmla}. \forall q \in \text{num}. \text{Fvars } \varphi = \{\text{zz}\} \wedge (\forall p \in \text{num}. \text{bprv } (\text{subst } \varphi \text{ } p \text{ } \text{zz}))$
 $\longrightarrow \text{prv } (\text{all } \text{zz } (\text{imp } (\text{LLq } (\text{Var } \text{zz}) \text{ } q) \varphi))$
and*

Lq_num2:
 let *LLq* = (λ *t1 t2. psubst Lq [(t1,zz), (t2,yy)]) in
 $\forall p \in \text{num}. \exists P \subseteq \text{num}. \text{finite } P \wedge \text{prv } (\text{dsj } (\text{sdsj } \{\text{eql } (\text{Var } \text{yy}) \text{ } r \mid r. r \in P\}) (\text{LLq } p \text{ } (\text{Var } \text{yy})))$
begin*

lemma *LLq_num*:
assumes $\varphi \in \text{fmla } q \in \text{num } \text{Fvars } \varphi = \{\text{zz}\} \forall p \in \text{num}. \text{bprv } (\text{subst } \varphi \text{ } p \text{ } \text{zz})$
shows $\text{prv } (\text{all } \text{zz } (\text{imp } (\text{LLq } (\text{Var } \text{zz}) \text{ } q) \varphi))$
 <proof>

lemma *LLq_num2*:
assumes $p \in \text{num}$
shows $\exists P \subseteq \text{num}. \text{finite } P \wedge \text{prv } (\text{dsj } (\text{sdsj } \{\text{eql } (\text{Var } \text{yy}) \text{ } r \mid r. r \in P\}) (\text{LLq } p \text{ } (\text{Var } \text{yy})))$
 <proof>

end — context *Deduct2_with_PseudoOrder*

1.2 Factoring In Explicit Proofs

```

locale Deduct_with_Proofs =
  Deduct_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv
+
fixes
proof :: 'proof set
and
prfOf :: 'proof  $\Rightarrow$  'fmla  $\Rightarrow$  bool
assumes
— Provability means there exists a proof (only needed for sentences):
prv_prfOf:  $\bigwedge \varphi. \varphi \in fmla \implies Fvars \varphi = \{\} \implies prv \varphi \longleftrightarrow (\exists prf \in proof. prfOf prf \varphi)$ 

```

```

locale Deduct2_with_Proofs =
  Deduct2_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv bprv
+
  Deduct_with_Proofs
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
  proof prfOf
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and proof :: 'proof set and prfOf

```

```

locale Deduct2_with_Proofs_PseudoOrder =
  Deduct2_with_Proofs
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi

```



```

fls
dsj
num
prv bprv
proof prfOf
+
Deduct2_with_PseudoOrder
var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi
fls
dsj
num
prv bprv
Lq
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and proof :: 'proof set and prfOf
and Lq

```

Chapter 2

Abstract Encoding

Here we simply fix some unspecified encoding functions: encoding formulas and proofs as numerals.

```
locale Encode =  
Syntax_with_Numerals  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and num  
+  
fixes enc :: 'fmla  $\Rightarrow$  'trm ( $\langle \_ \rangle$ )  
assumes  
enc[simp,introl]:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{enc } \varphi \in \text{num}$   
begin  
  
end — context Encode
```

Explicit proofs (encoded as numbers), needed only for the harder half of Goedel's first, and for both half's of Rosser's version; not needed in Goedel's second.

```
locale Encode_Proofs =  
Encode  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
  enc  
+  
Deduct2_with_Proofs  
  var trm fmla Var FvarsT substT Fvars subst  
  eql cnj imp all exi  
  fls  
  dsj  
  num  
  prv bprv  
  proof prfOf  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and num  
and eql cnj imp all exi  
and prv bprv  
and enc ( $\langle \_ \rangle$ )  
and fls dsj  
and proof :: 'proof set and prfOf
```

+
fixes *encPf* :: 'proof ⇒ 'trm
assumes
encPf[simp,intro!]: $\bigwedge pf. pf \in proof \implies encPf\ pf \in num$

Chapter 3

Representability Assumptions

Here we make assumptions about various functions or relations being representable.

3.1 Representability of Negation

The negation function `neg` is assumed to be representable by a two-variable formula `N`.

```
locale Repr_Neg =
  Deduct2_with_False
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  num
  prv bprv
+
  Encode
  var trm fmla Var FvarsT substT Fvars subst
  num
  enc
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and num
and prv bprv
and enc ( $\_$ )
+
fixes N :: 'fmla
assumes
  N[simp,intro!]: N  $\in$  fmla
and
  Fvars_N[simp]: Fvars N = {xx,yy}
and
  neg_implies_prv_N:
 $\bigwedge \varphi.$ 
  let NN = ( $\lambda$  t1 t2. psubst N [(t1,xx), (t2,yy)]) in
   $\varphi \in$  fmla  $\longrightarrow$  Fvars  $\varphi = \{\}$   $\longrightarrow$  bprv (NN  $\langle \varphi \rangle$   $\langle$  neg  $\varphi \rangle$ )
and
  N_unique:
 $\bigwedge \varphi.$ 
  let NN = ( $\lambda$  t1 t2. psubst N [(t1,xx), (t2,yy)]) in
```

$\varphi \in \text{fmla} \longrightarrow \text{Fvars } \varphi = \{\} \longrightarrow$
 $\text{bprv } (\text{all } yy \ (\text{all } yy' \$
 $\quad (\text{imp } (\text{cnj } (\text{NN } \langle \varphi \rangle (\text{Var } yy)) (\text{NN } \langle \varphi \rangle (\text{Var } yy'))))$
 $\quad (\text{eql } (\text{Var } yy) (\text{Var } yy'))))$)

begin

NN is a notation for the predicate that takes terms and returns corresponding instances of N, obtained by substituting its free variables with these terms. This is very convenient for reasoning, and will be done for all the representing formulas we will consider.

definition NN where $\text{NN} \equiv \lambda t1 t2. \text{psubst } N \ [((t1,xx), (t2,yy))]$

lemma NN_def2: $t1 \in \text{trm} \Longrightarrow t2 \in \text{trm} \Longrightarrow yy \notin \text{FvarsT } t1 \Longrightarrow$
 $\text{NN } t1 t2 = \text{subst } (\text{subst } N t1 xx) t2 yy$
 $\langle \text{proof} \rangle$

lemma NN_neg:

$\varphi \in \text{fmla} \Longrightarrow \text{Fvars } \varphi = \{\} \Longrightarrow \text{bprv } (\text{NN } \langle \varphi \rangle \langle \text{neg } \varphi \rangle)$
 $\langle \text{proof} \rangle$

lemma NN_unique:

assumes $\varphi \in \text{fmla} \ \text{Fvars } \varphi = \{\}$
shows $\text{bprv } (\text{all } yy \ (\text{all } yy' \$
 $\quad (\text{imp } (\text{cnj } (\text{NN } \langle \varphi \rangle (\text{Var } yy)) (\text{NN } \langle \varphi \rangle (\text{Var } yy'))))$
 $\quad (\text{eql } (\text{Var } yy) (\text{Var } yy'))))$)
 $\langle \text{proof} \rangle$

lemma NN[simp,intro]:

$t1 \in \text{trm} \Longrightarrow t2 \in \text{trm} \Longrightarrow \text{NN } t1 t2 \in \text{fmla}$
 $\langle \text{proof} \rangle$

lemma Fvars_NN[simp]: $t1 \in \text{trm} \Longrightarrow t2 \in \text{trm} \Longrightarrow yy \notin \text{FvarsT } t1 \Longrightarrow$
 $\text{Fvars } (\text{NN } t1 t2) = \text{FvarsT } t1 \cup \text{FvarsT } t2$
 $\langle \text{proof} \rangle$

lemma [simp]:

$m \in \text{num} \Longrightarrow n \in \text{num} \Longrightarrow \text{subst } (\text{NN } m (\text{Var } yy)) n yy = \text{NN } m n$
 $m \in \text{num} \Longrightarrow n \in \text{num} \Longrightarrow \text{subst } (\text{NN } m (\text{Var } yy')) n yy = \text{NN } m (\text{Var } yy')$
 $m \in \text{num} \Longrightarrow \text{subst } (\text{NN } m (\text{Var } yy')) (\text{Var } yy) yy' = \text{NN } m (\text{Var } yy)$
 $n \in \text{num} \Longrightarrow \text{subst } (\text{NN } (\text{Var } xx) (\text{Var } yy)) n xx = \text{NN } n (\text{Var } yy)$
 $n \in \text{num} \Longrightarrow \text{subst } (\text{NN } (\text{Var } xx) (\text{Var } xx')) n xx = \text{NN } n (\text{Var } xx')$
 $m \in \text{num} \Longrightarrow n \in \text{num} \Longrightarrow \text{subst } (\text{NN } m (\text{Var } xx')) n zz = \text{NN } m (\text{Var } xx')$
 $n \in \text{num} \Longrightarrow \text{subst } (\text{NN } n (\text{Var } yy)) (\text{Var } xx') yy = \text{NN } n (\text{Var } xx')$
 $m \in \text{num} \Longrightarrow n \in \text{num} \Longrightarrow \text{subst } (\text{NN } m (\text{Var } xx')) n xx' = \text{NN } m n$
 $\langle \text{proof} \rangle$

lemma NN_unique2:

assumes $[\text{simp}]: \varphi \in \text{fmla} \ \text{Fvars } \varphi = \{\}$

shows

$\text{bprv } (\text{all } yy \$
 $\quad (\text{imp } (\text{NN } \langle \varphi \rangle (\text{Var } yy))$
 $\quad (\text{eql } \langle \text{neg } \varphi \rangle (\text{Var } yy))))$

$\langle \text{proof} \rangle$

lemma NN_neg_unique:

assumes $[\text{simp}]: \varphi \in \text{fmla} \ \text{Fvars } \varphi = \{\}$

shows

$\text{bprv } (\text{imp } (\text{NN } \langle \varphi \rangle (\text{Var } yy))$

```

      (eql ⟨neg φ⟩ (Var yy)) (is bprv ?A)
⟨proof⟩

```

lemma *NN_exi_cnj*:

assumes $\varphi[simp]$: $\varphi \in fmla$ *Fvars* $\varphi = \{\}$ **and** $\chi[simp]$: $\chi \in fmla$

assumes *f*: *Fvars* $\chi = \{yy\}$

shows *bprv* (eqv (subst χ ⟨neg φ⟩ *yy*)
 (exi *yy* (cnj χ (NN ⟨φ⟩ (Var *yy*))))))

(is *bprv* (eqv ?A ?B))

⟨proof⟩

end — context *Repr_Neg*

3.2 Representability of Self-Substitution

Self-substitution is the function that takes a formula φ and returns $\phi[\langle\phi\rangle/xx]$ (for the fixed variable *xx*). This is all that will be needed for the diagonalization lemma.

locale *Repr_SelfSubst* =

Encode

```

  var trm fmla Var FvarsT substT Fvars subst
  num
  enc

```

+

Deduct2

```

  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv bprv

```

for

var :: 'var set **and** *trm* :: 'trm set **and** *fmla* :: 'fmla set

and *Var FvarsT substT Fvars subst*

and *num*

and *eql cnj imp all exi*

and *prv bprv*

and *enc* (⟨_⟩)

+

fixes *S* :: 'fmla

assumes

S[simp,intro!]: *S* ∈ *fmla*

and

Fvars_S[simp]: *Fvars S* = {*xx,yy*}

and

subst_implies_prv_S:

$\bigwedge \varphi.$

let *SS* = ($\lambda t1 t2. psubst S [(t1,xx), (t2,yy)]$) in

$\varphi \in fmla \longrightarrow Fvars \varphi = \{xx\} \longrightarrow$

bprv (*SS* ⟨ φ ⟩ ⟨subst φ ⟨ φ ⟩ *xx*)

and

S_unique:

$\bigwedge \varphi.$

let *SS* = ($\lambda t1 t2. psubst S [(t1,xx), (t2,yy)]$) in

$\varphi \in fmla \longrightarrow Fvars \varphi = \{xx\} \longrightarrow$

bprv (all *yy* (all *yy'*

(imp (cnj (*SS* ⟨ φ ⟩ (Var *yy*)) (*SS* ⟨ φ ⟩ (Var *yy'*))))

(eql (Var *yy*) (Var *yy'*))))))

begin

SS is the instantiation combinator of S:

definition *SS* **where** $SS \equiv \lambda t1\ t2.\ psubst\ S\ [(t1,xx), (t2,yy)]$

lemma *SS_def2*: $t1 \in trm \implies t2 \in trm \implies$
 $yy \notin FvarsT\ t1 \implies$
 $SS\ t1\ t2 = subst\ (subst\ S\ t1\ xx)\ t2\ yy$
 ⟨proof⟩

lemma *subst_implies_prv_SS*:
 $\varphi \in fmla \implies Fvars\ \varphi = \{xx\} \implies bprv\ (SS\ \langle\varphi\rangle\ \langle subst\ \varphi\ \langle\varphi\rangle\ xx\rangle)$
 ⟨proof⟩

lemma *SS_unique*:
 $\varphi \in fmla \implies Fvars\ \varphi = \{xx\} \implies$
 $bprv\ (all\ yy\ (all\ yy'\$
 $(imp\ (cnj\ (SS\ \langle\varphi\rangle\ (Var\ yy))\ (SS\ \langle\varphi\rangle\ (Var\ yy'))$
 $(eql\ (Var\ yy)\ (Var\ yy')))))$
 ⟨proof⟩

lemma *SS[simp,intro]*:
 $t1 \in trm \implies t2 \in trm \implies SS\ t1\ t2 \in fmla$
 ⟨proof⟩

lemma *Fvars_SS[simp]*: $t1 \in trm \implies t2 \in trm \implies yy \notin FvarsT\ t1 \implies$
 $Fvars\ (SS\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$
 ⟨proof⟩

lemma [*simp*]:
 $m \in num \implies p \in num \implies subst\ (SS\ m\ (Var\ yy))\ p\ yy = SS\ m\ p$
 $m \in num \implies subst\ (SS\ m\ (Var\ yy'))\ (Var\ yy)\ yy' = SS\ m\ (Var\ yy)$
 $m \in num \implies p \in num \implies subst\ (SS\ m\ (Var\ yy'))\ p\ yy' = SS\ m\ p$
 $m \in num \implies p \in num \implies subst\ (SS\ m\ (Var\ yy'))\ p\ yy = SS\ m\ (Var\ yy')$
 $m \in num \implies subst\ (SS\ (Var\ xx)\ (Var\ yy))\ m\ xx = SS\ m\ (Var\ yy)$
 ⟨proof⟩

end — context *Repr_SelfSubst*

3.3 Representability of Self-Soft-Substitution

The soft substitution function performs substitution logically instead of syntactically. In particular, its "self" version sends φ to $exi\ xx\ (cnj\ (eql\ (Var\ xx)\ (enc\ \varphi))\ \varphi)$. Representability of self-soft-substitution will be an alternative assumption in the diagonalization lemma.

locale *Repr_SelfSoftSubst* =
 Encode
 $var\ trm\ fmla\ Var\ FvarsT\ substT\ Fvars\ subst$
 num
 enc
 +
 Deduct2
 $var\ trm\ fmla\ Var\ FvarsT\ substT\ Fvars\ subst$
 num
 $eql\ cnj\ imp\ all\ exi$
 $prv\ bprv$
for
 $var :: 'var\ set\ \mathbf{and}\ trm :: 'trm\ set\ \mathbf{and}\ fmla :: 'fmla\ set$
and $Var\ FvarsT\ substT\ Fvars\ subst$

and *num*
and *eql conj imp all exi*
and *prv bprv*
and *enc* ($\langle _ \rangle$)
+
fixes $S :: 'fmla$
assumes
 $S[\text{simp, intro}] : S \in fmla$
and
 $Fvars_S[\text{simp}] : Fvars\ S = \{xx, yy\}$
and
 $\text{softSubst_implies_prv_}S :$
 $\bigwedge \varphi.$
 $\text{let } SS = (\lambda t1\ t2. \text{psubst } S [(t1, xx), (t2, yy)]) \text{ in}$
 $\varphi \in fmla \longrightarrow Fvars\ \varphi = \{xx\} \longrightarrow$
 $\text{bprv } (SS\ \langle \varphi \rangle\ \langle \text{softSubst } \varphi\ \langle \varphi \rangle\ xx)$
and
 $S_unique :$
 $\bigwedge \varphi.$
 $\text{let } SS = (\lambda t1\ t2. \text{psubst } S [(t1, xx), (t2, yy)]) \text{ in}$
 $\varphi \in fmla \longrightarrow Fvars\ \varphi = \{xx\} \longrightarrow$
 $\text{bprv } (\text{all } yy\ (\text{all } yy' \text{ (imp (conj (SS } \langle \varphi \rangle\ (\text{Var } yy)) (SS\ \langle \varphi \rangle\ (\text{Var } yy'))))$
 $\quad (\text{eql } (\text{Var } yy)\ (\text{Var } yy')))))$
begin

SS is the instantiation combinator of S:

definition SS **where** $SS \equiv \lambda t1\ t2. \text{psubst } S [(t1, xx), (t2, yy)]$

lemma $SS_def2 : t1 \in trm \implies t2 \in trm \implies$
 $yy \notin FvarsT\ t1 \implies$
 $SS\ t1\ t2 = \text{subst } (\text{subst } S\ t1\ xx)\ t2\ yy$
 $\langle \text{proof} \rangle$

lemma $\text{softSubst_implies_prv_}SS :$
 $\varphi \in fmla \implies Fvars\ \varphi = \{xx\} \implies \text{bprv } (SS\ \langle \varphi \rangle\ \langle \text{softSubst } \varphi\ \langle \varphi \rangle\ xx)$
 $\langle \text{proof} \rangle$

lemma $SS_unique :$
 $\varphi \in fmla \implies Fvars\ \varphi = \{xx\} \implies$
 $\text{bprv } (\text{all } yy\ (\text{all } yy' \text{ (imp (conj (SS } \langle \varphi \rangle\ (\text{Var } yy)) (SS\ \langle \varphi \rangle\ (\text{Var } yy'))))$
 $\quad (\text{eql } (\text{Var } yy)\ (\text{Var } yy')))))$
 $\langle \text{proof} \rangle$

lemma $SS[\text{simp, intro}] :$
 $t1 \in trm \implies t2 \in trm \implies SS\ t1\ t2 \in fmla$
 $\langle \text{proof} \rangle$

lemma $Fvars_SS[\text{simp}] : t1 \in trm \implies t2 \in trm \implies yy \notin FvarsT\ t1 \implies$
 $Fvars\ (SS\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$
 $\langle \text{proof} \rangle$

lemma $[\text{simp}] :$
 $m \in num \implies p \in num \implies \text{subst } (SS\ m\ (\text{Var } yy))\ p\ yy = SS\ m\ p$
 $m \in num \implies \text{subst } (SS\ m\ (\text{Var } yy'))\ (\text{Var } yy)\ yy' = SS\ m\ (\text{Var } yy)$
 $m \in num \implies p \in num \implies \text{subst } (SS\ m\ (\text{Var } yy'))\ p\ yy' = SS\ m\ p$
 $m \in num \implies p \in num \implies \text{subst } (SS\ m\ (\text{Var } yy'))\ p\ yy = SS\ m\ (\text{Var } yy')$

$m \in \text{num} \implies \text{subst } (SS \text{ (Var } xx) \text{ (Var } yy)) \ m \ xx = SS \ m \text{ (Var } yy)$
 $\langle \text{proof} \rangle$

end — context *Repr_SelfSoftSubst*

3.4 Clean Representability of the "Proof-of" Relation

For the proof-of relation, we must assume a stronger version of representability, namely clean representability on the first argument, which is dedicated to encoding the proof component. The property asks that the representation predicate is provably false on numerals that do not encode proofs; it would hold trivially for surjective proof encodings.

Cleanness is not a standard concept in the literature – we have introduced it in our CADE 2019 paper [1].

```

locale CleanRepr_Proofs =
  Encode_Proofs
    var trm fmla Var FvarsT substT Fvars subst
      num
      eql cnj imp all exi
      prv bprv
      enc
      fls
      dsj
      proof prfOf
      encPf
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and eql cnj imp all exi
and prv bprv
and enc ( $\langle \_ \rangle$ )
and fls dsj
and proof :: 'proof set and prfOf
and encPf
+
fixes Pf :: 'fmla
assumes
  Pf[simp,intro!]: Pf  $\in$  fmla
and
  Fvars_Pf[simp]: Fvars Pf = {yy,xx}
and
  prfOf_Pf:
   $\bigwedge$  prf  $\varphi$ .
    let PPf = ( $\lambda$  t1 t2. psubst Pf [(t1,yy), (t2,xx)]) in
    (prf  $\in$  proof  $\wedge$   $\varphi \in$  fmla  $\wedge$  Fvars  $\varphi = \{\}$ )  $\longrightarrow$ 
    prfOf prf  $\varphi$ 
     $\longrightarrow$ 
    bprv (PPf (encPf prf)  $\langle \varphi \rangle$ )
and
  not_prfOf_Pf:
   $\bigwedge$  prf  $\varphi$ .
    let PPf = ( $\lambda$  t1 t2. psubst Pf [(t1,yy), (t2,xx)]) in
    (prf  $\in$  proof  $\wedge$   $\varphi \in$  fmla  $\wedge$  Fvars  $\varphi = \{\}$ )  $\longrightarrow$ 
     $\neg$  prfOf prf  $\varphi$ 
     $\longrightarrow$ 
    bprv (neg (PPf (encPf prf)  $\langle \varphi \rangle$ ))

```

and

Clean_Pf_encPf:

$\wedge p \varphi$. let $PPf = (\lambda t1 t2. psubst Pf [(t1,yy), (t2,xx)])$ in
 $p \in num \wedge \varphi \in fmla \wedge Fvars \varphi = \{\}$ $\longrightarrow p \notin encPf \text{ ' proof } \longrightarrow bprv (neg (PPf p \langle \varphi \rangle))$

begin

PPf is the instantiation combinator of Pf:

definition *PPf* where $PPf \equiv \lambda t1 t2. psubst Pf [(t1,yy), (t2,xx)]$

lemma *prfOf_PPf:*

assumes $prf \in proof \varphi \in fmla Fvars \varphi = \{\}$ *prfOf* $prf \varphi$

shows $bprv (PPf (encPf prf) \langle \varphi \rangle)$

<proof>

lemma *not_prfOf_PPf:*

assumes $prf \in proof \varphi \in fmla Fvars \varphi = \{\} \neg prfOf prf \varphi$

shows $bprv (neg (PPf (encPf prf) \langle \varphi \rangle))$

<proof>

lemma *Clean_PPf_encPf:*

assumes $\varphi \in fmla Fvars \varphi = \{\}$ **and** $p \in num p \notin encPf \text{ ' proof}$

shows $bprv (neg (PPf p \langle \varphi \rangle))$

<proof>

lemma *PPf[simp,intro!]:* $t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow xx \notin FvarsT t1 \Longrightarrow PPf t1 t2 \in fmla$

<proof>

lemma *PPf_def2:* $t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow xx \notin FvarsT t1 \Longrightarrow$

$PPf t1 t2 = subst (subst Pf t1 yy) t2 xx$

<proof>

lemma *Fvars_PPf[simp]:*

$t1 \in trm \Longrightarrow t2 \in trm \Longrightarrow xx \notin FvarsT t1 \Longrightarrow$

$Fvars (PPf t1 t2) = FvarsT t1 \cup FvarsT t2$

<proof>

lemma [*simp*]:

$n \in num \Longrightarrow subst (PPf (Var yy) (Var xx)) n xx = PPf (Var yy) n$

$m \in num \Longrightarrow n \in num \Longrightarrow subst (PPf (Var yy) m) n yy = PPf n m$

$n \in num \Longrightarrow subst (PPf (Var yy) (Var xx)) n yy = PPf n (Var xx)$

$m \in num \Longrightarrow n \in num \Longrightarrow subst (PPf m (Var xx)) n xx = PPf m n$

$m \in num \Longrightarrow subst (PPf (Var zz) (Var xx')) m zz = PPf m (Var xx')$

$m \in num \Longrightarrow n \in num \Longrightarrow subst (PPf m (Var xx')) n xx' = PPf m n$

$n \in num \Longrightarrow subst (PPf (Var zz) (Var xx')) n xx' = PPf (Var zz) n$

$m \in num \Longrightarrow n \in num \Longrightarrow subst (PPf (Var zz) n) m zz = PPf m n$

<proof>

lemma *B_consistent_prfOf_iff_PPf:*

$B.consistent \Longrightarrow prf \in proof \Longrightarrow \varphi \in fmla \Longrightarrow Fvars \varphi = \{\} \longrightarrow prfOf prf \varphi \longleftrightarrow bprv (PPf (encPf prf) \langle \varphi \rangle)$

<proof>

lemma *consistent_prfOf_iff_PPf:*

$consistent \Longrightarrow prf \in proof \Longrightarrow \varphi \in fmla \Longrightarrow Fvars \varphi = \{\} \longrightarrow prfOf prf \varphi \longleftrightarrow bprv (PPf (encPf prf) \langle \varphi \rangle)$

<proof>

end — context *CleanRepr_Proofs*

Chapter 4

Diagonalization

This theory proves abstract versions of the diagonalization lemma, with both hard and soft substitution.

4.1 Alternative Diagonalization via Self-Substitution

Assuming representability of the diagonal instance of the substitution function, we prove the standard diagonalization lemma. More precisely, we show that it applies to any logic that – embeds intuitionistic first-order logic over numerals – has a countable number of formulas – has formula self-substitution representable

context *Repr_SelfSubst*
begin

theorem *diagonalization*:

assumes $\varphi[\text{simp, intro!}]$: $\varphi \in \text{fmla}$ $F\text{vars } \varphi = \{xx\}$
shows $\exists \psi. \psi \in \text{fmla} \wedge F\text{vars } \psi = \{\}$ \wedge $b\text{prv } (\text{equiv } \psi (\text{subst } \varphi \langle \psi \rangle xx))$
<proof>

Making this existential into a function.

definition *diag* :: $\text{'fmla} \Rightarrow \text{'fmla}$ **where**

$\text{diag } \varphi \equiv \text{SOME } \psi. \psi \in \text{fmla} \wedge F\text{vars } \psi = \{\}$ \wedge $b\text{prv } (\text{equiv } \psi (\text{subst } \varphi \langle \psi \rangle xx))$

theorem *diag_everything*:

assumes $\varphi \in \text{fmla}$ **and** $F\text{vars } \varphi = \{xx\}$
shows $\text{diag } \varphi \in \text{fmla} \wedge F\text{vars } (\text{diag } \varphi) = \{\}$ \wedge $b\text{prv } (\text{equiv } (\text{diag } \varphi) (\text{subst } \varphi \langle \text{diag } \varphi \rangle xx))$
<proof>

lemmas $\text{diag}[\text{simp}] = \text{diag_everything}[\text{THEN } \text{conjunct1}]$

lemmas $F\text{vars_diag}[\text{simp}] = \text{diag_everything}[\text{THEN } \text{conjunct2}, \text{THEN } \text{conjunct1}]$

lemmas $b\text{prv_diag_equiv} = \text{diag_everything}[\text{THEN } \text{conjunct2}, \text{THEN } \text{conjunct2}]$

end — context *Repr_SelfSubst*

4.2 Alternative Diagonalization via Soft Self-Substitution

context *Repr_SelfSoftSubst*
begin

theorem *diagonalization*:

assumes $\varphi[\text{simp, intro!}]$: $\varphi \in \text{fmla}$ $F\text{vars } \varphi = \{xx\}$

shows $\exists \psi. \psi \in fmla \wedge Fvars \psi = \{\} \wedge bprv (eqv \psi (subst \varphi \langle \psi \rangle xx))$
<proof>

Making this existential into a function.

definition *diag* :: 'fmla \Rightarrow 'fmla **where**

diag $\varphi \equiv SOME \psi. \psi \in fmla \wedge Fvars \psi = \{\} \wedge bprv (eqv \psi (subst \varphi \langle \psi \rangle xx))$

theorem *diag_everything*:

assumes $\varphi \in fmla$ **and** $Fvars \varphi = \{xx\}$

shows $diag \varphi \in fmla \wedge Fvars (diag \varphi) = \{\} \wedge bprv (eqv (diag \varphi) (subst \varphi \langle diag \varphi \rangle xx))$

<proof>

lemmas *diag[simp]* = *diag_everything*[*THEN* *conjunct1*]

lemmas *Fvars_diag[simp]* = *diag_everything*[*THEN* *conjunct2*, *THEN* *conjunct1*]

lemmas *prv_diag_eqv* = *diag_everything*[*THEN* *conjunct2*, *THEN* *conjunct2*]

end — context *Repr_SelfSoftSubst*

Chapter 5

The Hilbert-Bernays-Löb (HBL) Derivability Conditions

5.1 First Derivability Condition

```
locale HBL1 =
  Encode
    var trm fmla Var FvarsT substT Fvars subst
    num
  enc
+
  Deduct2
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and eql cnj imp all exi
and prv bprv
and enc (⟨_⟩)
+

  fixes P :: 'fmla
  assumes
  P[intro!,simp]: P ∈ fmla
and
  Fvars_PP[simp]: Fvars P = {xx}
and
  HBL1:  $\bigwedge \varphi. \varphi \in fmla \implies Fvars \varphi = \{\} \implies prv \varphi \implies bprv (subst P \langle \varphi \rangle xx)$ 
begin

definition PP where PP  $\equiv \lambda t. subst P t xx$ 

lemma PP[simp]:  $\bigwedge t. t \in trm \implies PP t \in fmla$ 
  ⟨proof⟩

lemma Fvars_PP[simp]:  $\bigwedge t. t \in trm \implies Fvars (PP t) = FvarsT t$ 
  ⟨proof⟩
```

lemma *[simp]*:
 $n \in \text{num} \implies \text{subst } (PP \text{ (Var } yy)) \ n \ yy = PP \ n$
 $n \in \text{num} \implies \text{subst } (PP \text{ (Var } xx)) \ n \ xx = PP \ n$
 ⟨*proof*⟩

lemma *HBL1_PP*:
 $\varphi \in \text{fm}la \implies \text{Fvars } \varphi = \{\} \implies \text{prv } \varphi \implies \text{bprv } (PP \ \langle\varphi\rangle)$
 ⟨*proof*⟩

end — context *HBL1*

5.2 Connections between Proof Representability, First Derivability Condition, and Its Converse

context *CleanRepr_Proofs*
begin

Defining P , the internal notion of provability, from Pf (in its predicate form PPf), the internal notion of "proof-of". NB: In the technical sense of the term "represents", we have that Pf represents pprv , whereas P will not represent prv , but satisfy a weaker condition (weaker than weak representability), namely HBL1.

5.2.1 HBL1 from "proof-of" representability

definition $P :: 'fm}la \text{ where } P \equiv \text{exi } yy \ (PPf \text{ (Var } yy) \text{ (Var } xx))$

lemma $P[\text{simp}, \text{intro}]$: $P \in \text{fm}la \text{ and } \text{Fvars_}P[\text{simp}]$: $\text{Fvars } P = \{xx\}$
 ⟨*proof*⟩

We infer HBL1 from Pf representing prv :

lemma *HBL1*:
assumes $\varphi \in \text{fm}la \ \text{Fvars } \varphi = \{\}$ **and** $p\varphi$: $\text{prv } \varphi$
shows $\text{bprv } (\text{subst } P \ \langle\varphi\rangle \ xx)$
 ⟨*proof*⟩

This is used in several places, including for the hard half of Gödel's First and the truth of Gödel formulas (and also for the Rosser variants of these).

lemma *not_prv_prv_neg_PPf*:
assumes $[\text{simp}]$: $\varphi \in \text{fm}la \ \text{Fvars } \varphi = \{\}$ **and** p : $\neg \text{prv } \varphi$ **and** $n[\text{simp}]$: $n \in \text{num}$
shows $\text{bprv } (\text{neg } (PPf \ n \ \langle\varphi\rangle))$
 ⟨*proof*⟩

lemma *consistent_not_prv_not_prv_PPf*:
assumes c : *consistent*
and $\theta[\text{simp}]$: $\varphi \in \text{fm}la \ \text{Fvars } \varphi = \{\} \ \neg \text{prv } \varphi \ n \in \text{num}$
shows $\neg \text{bprv } (PPf \ n \ \langle\varphi\rangle)$
 ⟨*proof*⟩

end — context *CleanRepr_Proofs*

The inference of HBL1 from "proof-of" representability, in locale form:

sublocale *CleanRepr_Proofs* < *wrepr*: *HBL1*
where $P = P$
 ⟨*proof*⟩

5.2.2 Sufficient condition for the converse of HBL1

context *CleanRepr_Proofs*
begin

lemma *PP_PPf*:
assumes $\varphi \in \text{fmla}$
shows $\text{wrepr.PP } \langle \varphi \rangle = \text{exi yy } (\text{PPf } (\text{Var yy}) \langle \varphi \rangle)$
 $\langle \text{proof} \rangle$

The converse of HBL1 condition follows from (the standard notion of) ω -consistency for *bprv* and strong representability of proofs:

lemma *ω consistentStd1_HBL1_rev*:
assumes *oc*: $B.\omega\text{consistentStd1}$
and $\varphi[\text{simp}]$: $\varphi \in \text{fmla } F\text{vars } \varphi = \{\}$
and *iPP*: $\text{bprv } (\text{wrepr.PP } \langle \varphi \rangle)$
shows *prv* φ
 $\langle \text{proof} \rangle$

end — context *CleanRepr_Proofs*

5.3 Second and Third Derivability Conditions

These are only needed for Gödel's Second.

locale *HBL1_2_3* =
HBL1
 $\text{var trm fmla Var FvarsT substT Fvars subst}$
 num
 $\text{eql cnj imp all exi}$
 prv bprv
 enc
 P
for
 $\text{var} :: \text{'var set and trm} :: \text{'trm set and fmla} :: \text{'fmla set}$
and $\text{Var FvarsT substT Fvars subst}$
and num
and $\text{eql cnj imp all exi}$
and prv bprv
and $\text{enc } (\langle _ \rangle)$
and P
 $+$
assumes
HBL2: $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies F\text{vars } \varphi = \{\} \implies F\text{vars } \chi = \{\} \implies$
 $\text{bprv } (\text{imp } (\text{cnj } (\text{PP } \langle \varphi \rangle) (\text{PP } \langle \text{imp } \varphi \chi \rangle)))$
 $(\text{PP } \langle \chi \rangle)$
and
HBL3: $\bigwedge \varphi. \varphi \in \text{fmla} \implies F\text{vars } \varphi = \{\} \implies \text{bprv } (\text{imp } (\text{PP } \langle \varphi \rangle) (\text{PP } \langle \text{PP } \langle \varphi \rangle \rangle))$
begin

The implicational form of HBL2:

lemma *HBL2_imp*:
 $\bigwedge \varphi \chi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies F\text{vars } \varphi = \{\} \implies F\text{vars } \chi = \{\} \implies$
 $\text{bprv } (\text{imp } (\text{PP } \langle \text{imp } \varphi \chi \rangle) (\text{imp } (\text{PP } \langle \varphi \rangle) (\text{PP } \langle \chi \rangle)))$
 $\langle \text{proof} \rangle$

... and its weaker, "detached" version:

lemma *HBL2_imp2*:


```
assumes  $\varphi \in \text{fmla}$  and  $\chi \in \text{fmla}$   $Fvars\ \varphi = \{\}$   $Fvars\ \chi = \{\}$   
assumes  $bprv\ (PP\ \langle imp\ \varphi\ \chi \rangle)$   
shows  $bprv\ (imp\ (PP\ \langle \varphi \rangle)\ (PP\ \langle \chi \rangle))$   
 $\langle proof \rangle$   
end — context  $HBL1\_2\_3$ 
```

Chapter 6

Gödel Formulas

Gödel formulas are defined by diagonalizing the negation of the provability predicate.

```
locale Goedel_Form =  
— Assuming the fls (False) connective gives us negation:  
Deduct2_with_False  
  var trm fmla Var FvarsT substT Fvars subst  
  eql cnj imp all exi  
  fls  
  num  
  prv bprv  
+  
Repr_SelfSubst  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
  eql cnj imp all exi  
  prv bprv  
  enc  
  S  
+  
HBL1  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
  eql cnj imp all exi  
  prv bprv  
  enc  
  P  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var num FvarsT substT Fvars subst  
and eql cnj imp all exi  
and fls  
and prv bprv  
and enc ( $\langle \_ \rangle$ )  
and S  
and P  
begin
```

The Gödel formula. NB, we speak of "the" Gödel formula because the diagonalization function makes a choice.

definition φG :: '*fmla* **where** $\varphi G \equiv \text{diag } (\text{neg } P)$

lemma $\varphi G[\text{simp, intro}]$: $\varphi G \in \text{fmla}$
and

Fvars_φG[simp]: Fvars φG = {}
 ⟨proof⟩

lemma *bprv_φG_eqv:*
bprv (eqv φG (neg (PP ⟨φG⟩)))
 ⟨proof⟩

lemma *prv_φG_eqv:*
prv (eqv φG (neg (PP ⟨φG⟩)))
 ⟨proof⟩

end — context *Goedel_Form*

Adding cleanly representable proofs to the assumptions behind Gödel formulas:

locale *Goedel_Form_Proofs =*
Repr_SelfSubst
var trm fmla Var FvarsT substT Fvars subst
num
eql cnj imp all exi
prv bprv
enc
S
 +
CleanRepr_Proofs
var trm fmla Var FvarsT substT Fvars subst
num
eql cnj imp all exi
prv bprv
enc
fls
dsj
proof prfOf
encPf
Pf
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and *Var FvarsT substT Fvars subst num*
and *eql cnj imp all exi*
and *fls*
and *prv bprv*
and *enc (⟨_⟩)*
and *S*
and *dsj*
and *proof :: 'proof set and prfOf encPf*
and *Pf*

... and extending the sublocale relationship *CleanRepr_Proofs < HBL1*:

sublocale *Goedel_Form_Proofs < Goedel_Form where P = P ⟨proof⟩*

context *Goedel_Form_Proofs*
begin

lemma *bprv_φG_eqv_not_exi_PPf:*
bprv (eqv φG (neg (exi yy (PPf (Var yy) ⟨φG⟩))))
 ⟨proof⟩

lemma *prv_φG_eqv_not_exi_PPf:*

prv (*eqv* φG (*neg* (*exi* *yy* (*PPf* (*Var* *yy*) $\langle \varphi G \rangle$))))
<proof>

lemma *bprv_φG_eqv_all_not_PPf*:
bprv (*eqv* φG (*all* *yy* (*neg* (*PPf* (*Var* *yy*) $\langle \varphi G \rangle$))))
<proof>

lemma *prv_φG_eqv_all_not_PPf*:
prv (*eqv* φG (*all* *yy* (*neg* (*PPf* (*Var* *yy*) $\langle \varphi G \rangle$))))
<proof>

lemma *bprv_eqv_all_not_PPf_imp_φG*:
bprv (*imp* (*all* *yy* (*neg* (*PPf* (*Var* *yy*) $\langle \varphi G \rangle$))) φG)
<proof>

lemma *prv_eqv_all_not_PPf_imp_φG*:
prv (*imp* (*all* *yy* (*neg* (*PPf* (*Var* *yy*) $\langle \varphi G \rangle$))) φG)
<proof>

end — context *Goedel_Form_Proofs*

Chapter 7

Standard Models with Two Provability Relations

```
locale Minimal_Truth_Soundness_Proof_Repr =
  CleanRepr_Proofs
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv bprv
  enc
  fls
  dsj
  proof prfOf
  encPf
  Pf
+ — The label "B" stands for "basic", as a reminder that soundness refers to the basic provability relation:
B: Minimal_Truth_Soundness
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  bprv
  isTrue
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
  and Var FvarsT substT Fvars subst
  and eql cnj imp all exi
  and fls
  and dsj
  and num
  and prv bprv
  and isTrue
  and enc (⟦_⟧)
  and proof :: 'proof set and prfOf
  and encPf Pf
begin

lemmas prfOf_iff_PPf = B_consistent_prfOf_iff_PPf[OF B.consistent]

The provability predicate is decided by basic provability on encodings:
lemma isTrue_prv_PPf_prf_or_neg:
```

$prf \in proof \implies \varphi \in fmla \implies Fvars \varphi = \{\} \implies$
 $bprv (PPf (encPf prf) \langle \varphi \rangle) \vee bprv (neg (PPf (encPf prf) \langle \varphi \rangle))$
 $\langle proof \rangle$

... hence that predicate is complete w.r.t. truth:

lemma *isTrue_PPf_Implies_bprv_PPf*:
 $prf \in proof \implies \varphi \in fmla \implies Fvars \varphi = \{\} \implies$
 $isTrue (PPf (encPf prf) \langle \varphi \rangle) \implies bprv (PPf (encPf prf) \langle \varphi \rangle)$
 $\langle proof \rangle$

... and thanks cleanness we can replace encoded proofs with arbitrary numerals in the completeness property:

lemma *isTrue_implies_bprv_PPf*:
assumes *[simp]*: $n \in num \varphi \in fmla Fvars \varphi = \{\}$
and *iT*: $isTrue (PPf n \langle \varphi \rangle)$
shows $bprv (PPf n \langle \varphi \rangle)$
 $\langle proof \rangle$

... in fact, by *Minimal_Truth_Soundness* we even have an iff:

lemma *isTrue_iff_bprv_PPf*:
 $\wedge n \varphi. n \in num \implies \varphi \in fmla \implies Fvars \varphi = \{\} \implies isTrue (PPf n \langle \varphi \rangle) \longleftrightarrow bprv (PPf n \langle \varphi \rangle)$
 $\langle proof \rangle$

Truth of the provability representation implies provability (TIP):

lemma *TIP*:
assumes $\varphi[simp]$: $\varphi \in fmla Fvars \varphi = \{\}$
and *iPP*: $isTrue (wrepr.PP \langle \varphi \rangle)$
shows $prv \varphi$
 $\langle proof \rangle$

The reverse HBL1 – now without the ω -consistency assumption which holds here thanks to our truth-in-standard-model assumption:

lemmas *HBL1_rev = ω consistentStd1_HBL1_rev*[OF *B. ω consistentStd1*]

Note that the above would also follow by *Minimal_Truth_Soundness* from TIP:

lemma *TIP_implies_HBL1_rev*:
assumes *TIP*: $\forall \varphi. \varphi \in fmla \wedge Fvars \varphi = \{\} \wedge isTrue (wrepr.PP \langle \varphi \rangle) \longrightarrow prv \varphi$
shows $\forall \varphi. \varphi \in fmla \wedge Fvars \varphi = \{\} \wedge bprv (wrepr.PP \langle \varphi \rangle) \longrightarrow prv \varphi$
 $\langle proof \rangle$

end — context *Minimal_Truth_Soundness_Proof_Repr*

7.1 Proof recovery from *HBL1_iff*

locale *Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf* =
HBL1

var trm fmla Var FvarsT substT Fvars subst
num
eql cnj imp all exi
prv bprv
enc
P

+

B : Minimal_Truth_Soundness
var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi

```

  fls
  dsj
  num
  bprv
  isTrue
+
Deduct_with_False_Disj
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and enc (⟨_⟩)
and prv bprv
and P
and isTrue
+
fixes Pf :: 'fmla
assumes
— Pf is a formula with free variables xx yy:
Pf[simp,intro!]: Pf ∈ fmla
and
Fvars_Pf[simp]: Fvars Pf = {yy,xx}
and
— P relates to Pf internally (inside basic provability) just like a prv and a prfOf would relate—via an
existential:
P_Pf:
φ ∈ fmla ⇒ Fvars φ = {} ⇒
  let PPf = (λ t1 t2. psubst Pf [(t1,yy), (t2,xx)]) in
  bprv (eql (subst P ⟨φ⟩ xx) (exi yy (PPf (Var yy) ⟨φ⟩)))
assumes
— We assume both HBL1 and HBL1_rev, i.e., an iff version:
HBL1_iff: ∧ φ. φ ∈ fmla ⇒ Fvars φ = {} ⇒ bprv (PP ⟨φ⟩) ↔ prv φ
and
Compl_Pf:
∧ n φ. n ∈ num ⇒ φ ∈ fmla ⇒ Fvars φ = {} ⇒
  let PPf = (λ t1 t2. psubst Pf [(t1,yy), (t2,xx)]) in
  isTrue (PPf n ⟨φ⟩) → bprv (PPf n ⟨φ⟩)
begin

definition PPf where PPf ≡ λ t1 t2. psubst Pf [(t1,yy), (t2,xx)]

lemma PP_def: PP t = subst P t xx ⟨proof⟩

lemma PP_PPf_eqv:
  φ ∈ fmla ⇒ Fvars φ = {} ⇒ bprv (eql (PP ⟨φ⟩) (exi yy (PPf (Var yy) ⟨φ⟩)))
  ⟨proof⟩

```

lemma *PPf[simp,intro!]*: $t1 \in trm \implies t2 \in trm \implies xx \notin FvarsT\ t1 \implies PPf\ t1\ t2 \in fmla$
 ⟨proof⟩

lemma *PPf_def2*: $t1 \in trm \implies t2 \in trm \implies xx \notin FvarsT\ t1 \implies$
 $PPf\ t1\ t2 = subst\ (subst\ Pf\ t1\ yy)\ t2\ xx$
 ⟨proof⟩

lemma *Fvars_PPf[simp]*:
 $t1 \in trm \implies t2 \in trm \implies xx \notin FvarsT\ t1 \implies Fvars\ (PPf\ t1\ t2) = FvarsT\ t1 \cup FvarsT\ t2$
 ⟨proof⟩

lemma [*simp*]:
 $n \in num \implies subst\ (PPf\ (Var\ yy)\ (Var\ xx))\ n\ xx = PPf\ (Var\ yy)\ n$
 $m \in num \implies n \in num \implies subst\ (PPf\ (Var\ yy)\ m)\ n\ yy = PPf\ n\ m$
 $n \in num \implies subst\ (PPf\ (Var\ yy)\ (Var\ xx))\ n\ yy = PPf\ n\ (Var\ xx)$
 $m \in num \implies n \in num \implies subst\ (PPf\ m\ (Var\ xx))\ n\ xx = PPf\ m\ n$
 $m \in num \implies subst\ (PPf\ (Var\ zz)\ (Var\ xx'))\ m\ zz = PPf\ m\ (Var\ xx')$
 $m \in num \implies n \in num \implies subst\ (PPf\ m\ (Var\ xx'))\ n\ xx' = PPf\ m\ n$
 $n \in num \implies subst\ (PPf\ (Var\ zz)\ (Var\ xx'))\ n\ xx' = PPf\ (Var\ zz)\ n$
 $m \in num \implies n \in num \implies subst\ (PPf\ (Var\ zz)\ n)\ m\ zz = PPf\ m\ n$
 ⟨proof⟩

lemma *PP_PPf*:
assumes $\varphi \in fmla\ Fvars\ \varphi = \{\}$ **shows** $bprv\ (PP\ \langle\varphi\rangle) \longleftrightarrow bprv\ (exi\ yy\ (PPf\ (Var\ yy)\ \langle\varphi\rangle))$
 ⟨proof⟩

lemma *isTrue_implies_bprv_PPf*:
 $\bigwedge n\ \varphi.\ n \in num \implies \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies$
 $isTrue\ (PPf\ n\ \langle\varphi\rangle) \implies bprv\ (PPf\ n\ \langle\varphi\rangle)$
 ⟨proof⟩

lemma *isTrue_iff_bprv_PPf*:
 $\bigwedge n\ \varphi.\ n \in num \implies \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies isTrue\ (PPf\ n\ \langle\varphi\rangle) \longleftrightarrow bprv\ (PPf\ n\ \langle\varphi\rangle)$
 ⟨proof⟩

Preparing to instantiate this "proof recovery" alternative into our mainstream locale hierarchy, which assumes proofs. We define the "missing" proofs to be numerals, we encode them as the identity, and we "copy" *prfOf* from the corresponding predicate one-level-up, *PPf*:

definition *proof* :: 'trm set **where** [*simp*]: $proof = num$

definition *prfOf* :: 'trm \Rightarrow 'fmla \Rightarrow bool **where**

$prfOf\ n\ \varphi \equiv bprv\ (PPf\ n\ \langle\varphi\rangle)$

definition *encPf* :: 'trm \Rightarrow 'trm **where** [*simp*]: $encPf \equiv id$

lemma *prv_exi_PPf_iff_isTrue*:

assumes [*simp*]: $\varphi \in fmla\ Fvars\ \varphi = \{\}$

shows $bprv\ (exi\ yy\ (PPf\ (Var\ yy)\ \langle\varphi\rangle)) \longleftrightarrow isTrue\ (exi\ yy\ (PPf\ (Var\ yy)\ \langle\varphi\rangle))$ (**is** ?L \longleftrightarrow ?R)
 ⟨proof⟩

lemma *isTrue_exi_iff*:

assumes [*simp*]: $\varphi \in fmla\ Fvars\ \varphi = \{\}$

shows $isTrue\ (exi\ yy\ (PPf\ (Var\ yy)\ \langle\varphi\rangle)) \longleftrightarrow (\exists n \in num.\ isTrue\ (PPf\ n\ \langle\varphi\rangle))$ (**is** ?L \longleftrightarrow ?R)
 ⟨proof⟩

lemma *prv_prfOf*:

assumes $\varphi \in fmla\ Fvars\ \varphi = \{\}$

shows $prv \varphi \longleftrightarrow (\exists n \in num. prfOf\ n \ \varphi)$
 ⟨proof⟩

lemma *prfOf_prv_Pf*:
assumes $n \in num$ **and** $\varphi \in fmla\ Fvars\ \varphi = \{\}$ **and** $prfOf\ n\ \varphi$
shows $bprv\ (psubst\ Pf\ [(n, yy), (\langle\varphi\rangle, xx)])$
 ⟨proof⟩

lemma *isTrue_exi_iff_PP*:
assumes $[\text{simp}]: \varphi \in fmla\ Fvars\ \varphi = \{\}$
shows $isTrue\ (PP\ \langle\varphi\rangle) \longleftrightarrow (\exists n \in num. isTrue\ (PPf\ n\ \langle\varphi\rangle))$
 ⟨proof⟩

lemma *bprv_compl_isTrue_PP_enc*:
assumes $1: \varphi \in fmla\ Fvars\ \varphi = \{\}$ **and** $2: isTrue\ (PP\ \langle\varphi\rangle)$
shows $bprv\ (PP\ \langle\varphi\rangle)$
 ⟨proof⟩

lemma *TIP*:
assumes $1: \varphi \in fmla\ Fvars\ \varphi = \{\}$ **and** $2: isTrue\ (PP\ \langle\varphi\rangle)$
shows $prv\ \varphi$
 ⟨proof⟩

end — context *Minimal_Truth_Soundness_HBL1iff_Compl_Pf*

locale *Minimal_Truth_Soundness_HBL1iff_Compl_Pf_Compl_NegPf* =
Minimal_Truth_Soundness_HBL1iff_Compl_Pf
 +

assumes
Compl_NegPf:
 $\bigwedge n\ \varphi. n \in num \implies \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies$
 $let\ PPf = (\lambda\ t1\ t2. psubst\ Pf\ [(t1, yy), (t2, xx)])\ in$
 $isTrue\ (B.neg\ (PPf\ n\ \langle\varphi\rangle)) \implies bprv\ (B.neg\ (PPf\ n\ \langle\varphi\rangle))$
begin

lemma *isTrue_implies_prv_neg_PPf*:
 $\bigwedge n\ \varphi. n \in num \implies \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies$
 $isTrue\ (B.neg\ (PPf\ n\ \langle\varphi\rangle)) \implies bprv\ (B.neg\ (PPf\ n\ \langle\varphi\rangle))$
 ⟨proof⟩

lemma *isTrue_iff_prv_neg_PPf*:
 $\bigwedge n\ \varphi. n \in num \implies \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies isTrue\ (B.neg\ (PPf\ n\ \langle\varphi\rangle)) \longleftrightarrow bprv\ (B.neg\ (PPf\ n\ \langle\varphi\rangle))$
 ⟨proof⟩

lemma *prv_PPf_decide*:
assumes $[\text{simp}]: n \in num\ \varphi \in fmla\ Fvars\ \varphi = \{\}$
and $np: \neg bprv\ (PPf\ n\ \langle\varphi\rangle)$
shows $bprv\ (B.neg\ (PPf\ n\ \langle\varphi\rangle))$
 ⟨proof⟩

lemma *not_prfOf_prv_neg_Pf*:
assumes $np: n \in num\ \varphi \in fmla\ Fvars\ \varphi = \{\}$ **and** $\neg prfOf\ n\ \varphi$
shows $bprv\ (B.neg\ (psubst\ Pf\ [(n, yy), (\langle\varphi\rangle, xx)]))$
 ⟨proof⟩

end — context *Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf*

sublocale *Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf* <
 repr: *CleanRepr_Proofs*

where *proof* = *proof* **and** *prfOf* = *prfOf* **and** *encPf* = *encPf*
 ⟨*proof*⟩

sublocale *Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf* <
 min_truth: *Minimal_Truth_Soundness_Proof_Repr*

where *proof* = *proof* **and** *prfOf* = *prfOf* **and** *encPf* = *encPf*
 ⟨*proof*⟩

locale *Minimal_Truth_Soundness_HBL1iff_prv_Cmpl_Pf* =
HBL1

 var *trm fmla Var FvarsT substT Fvars subst*
 num
 eql cnj imp all exi
 prv bprv
 enc
 P

+

B: Minimal_Truth_Soundness

 var *trm fmla Var FvarsT substT Fvars subst*
 eql cnj imp all exi
 fls
 dsj
 num
 bprv
 isTrue

+

Deduct_with_False_Disj

 var *trm fmla Var FvarsT substT Fvars subst*
 eql cnj imp all exi
 fls
 dsj
 num
 prv

for

var :: '*var set* **and** *trm* :: '*trm set* **and** *fmla* :: '*fmla set*

and *Var FvarsT substT Fvars subst*

and *eql cnj imp all exi*

and *fls*

and *dsj*

and *num*

and *enc* (⟨*_*⟩)

and *prv bprv*

and *P*

and *isTrue*

+

fixes *Pf* :: '*fmla*

assumes

Pf[*simp,intro!*]: $Pf \in \text{fm}la$

and

Fvars_Pf[*simp*]: $Fvars\ Pf = \{yy,xx\}$

and

P_Pf:

$\varphi \in \text{fm}la \implies$

let *PPf* = $(\lambda\ t1\ t2.\ \text{psubst}\ Pf\ [(t1,yy), (t2,xx)])$ *in*

bprv (*eqv* (*subst* *P* $\langle\varphi\rangle\ xx$) (*exi* *yy* (*PPf* (*Var* *yy*) $\langle\varphi\rangle$)))

assumes

HBL1_rev_prv: $\bigwedge\ \varphi.\ \varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies \text{prv}\ (PP\ \langle\varphi\rangle) \implies \text{prv}\ \varphi$

and

Compl_Pf:

$\bigwedge\ n\ \varphi.\ n \in \text{num} \implies \varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies$

let *PPf* = $(\lambda\ t1\ t2.\ \text{psubst}\ Pf\ [(t1,yy), (t2,xx)])$ *in*

isTrue (*PPf* *n* $\langle\varphi\rangle$) \longrightarrow *bprv* (*PPf* *n* $\langle\varphi\rangle$)

begin

lemma *HBL1_rev*:

assumes *f*: $\varphi \in \text{fm}la$ **and** *fv*: $Fvars\ \varphi = \{\}$ **and** *bp*: *bprv* (*PP* $\langle\varphi\rangle$)

shows *prv* φ

<proof>

lemma *HBL1_iff*: $\varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies \text{bprv}\ (PP\ \langle\varphi\rangle) \longleftrightarrow \text{prv}\ \varphi$

<proof>

lemma *HBL1_iff_prv*: $\varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies \text{prv}\ (PP\ \langle\varphi\rangle) \longleftrightarrow \text{prv}\ \varphi$

<proof>

end — context *Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf*

sublocale *Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf* <

mts_prv_mts: *Minimal_Truth_Soundness_HBL1iff_Compl_Pf* **where** *Pf* = *Pf*

<proof>

locale *Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical* =

Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf

+

assumes

— NB: we don't really need to assume classical reasoning (double negation) all throughout, but only for the provability predicate:

classical_P: $\bigwedge\ \varphi.\ \varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies \text{let}\ PP = (\lambda\ t.\ \text{subst}\ P\ t\ xx)$ *in*

prv (*B.neg* (*B.neg* (*PP* $\langle\varphi\rangle$))) \implies *prv* (*PP* $\langle\varphi\rangle$)

begin

lemma *classical_PP*: $\varphi \in \text{fm}la \implies Fvars\ \varphi = \{\} \implies \text{prv}\ (B.\text{neg}\ (B.\text{neg}\ (PP\ \langle\varphi\rangle))) \implies \text{prv}\ (PP\ \langle\varphi\rangle)$

<proof>

end — context *Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical*

Chapter 8

Abstract Formulations of Gödel's First Incompleteness Theorem

8.1 Proof-Theoretic Versions of Gödel's First

context *Goedel_Form*
begin

8.1.1 The easy half

First the "direct", positive formulation:

lemma *goedel_first_theEasyHalf_pos*:
assumes *prv* φG **shows** *prv* *fls*
(*proof*)

... then the more standard contrapositive formulation:

corollary *goedel_first_theEasyHalf*:
consistent $\implies \neg \text{prv } \varphi G$
(*proof*)

end — context *Goedel_Form*

8.1.2 The hard half

The hard half needs explicit proofs:

context *Goedel_Form_Proofs* **begin**

lemma *goedel_first_theHardHalf*:
assumes *oc*: $\omega\text{consistent}$
shows $\neg \text{prv } (\text{neg } \varphi G)$
(*proof*)

theorem *goedel_first*:
assumes $\omega\text{consistent}$
shows $\neg \text{prv } \varphi G \wedge \neg \text{prv } (\text{neg } \varphi G)$
(*proof*)

theorem *goedel_first_ex*:
assumes $\omega\text{consistent}$
shows $\exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi)$
(*proof*)

end — context *Goedel_Form_Proofs*

8.2 Model-Theoretic Versions of Gödel's First

The model-theoretic twist is that of additionally proving the truth of Gödel sentences.

8.2.1 First model-theoretic version

```

locale Goedel_Form_Proofs_Minimal_Truth =
  Goedel_Form_Proofs
    var trm fmla Var FvarsT substT Fvars subst
      num
      eql cnj imp all exi
      fls
      prv bprv
      enc
      S
      dsj
      proof prfOf encPf
      Pf
  +
  Minimal_Truth_Soundness
    var trm fmla Var FvarsT substT Fvars subst
      eql cnj imp all exi
      fls
      dsj
      num
      bprv
      isTrue
  for
    var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
    and Var FvarsT substT Fvars subst
    and eql cnj imp all exi
    and fls
    and dsj
    and num
    and prv bprv
    and enc ( $\langle \_ \rangle$ )
    and S
    and proof :: 'proof set and prfOf encPf
    and Pf
    and isTrue
  begin

```

Recall that "consistent" and " ω -consistent" refer to *prv*, not to *bprv*.

```

theorem isTrue_φG:
  assumes consistent
  shows isTrue φG
  ⟨proof⟩

```

The "strong" form of Gödel's First (also asserting the truth of the Gödel sentences):

```

theorem goedel_first_strong:
  ωconsistent  $\implies \neg prv \varphi G \wedge \neg prv (neg \varphi G) \wedge isTrue \varphi G$ 
  ⟨proof⟩

```

theorem *goedel_first_strong_ex*:
 $\omega\text{consistent} \implies \exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi) \wedge \text{isTrue } \varphi$
 ⟨proof⟩

end — context *Goedel_Form_Proofs_Minimal_Truth*

8.2.2 Second model-theoretic version

locale *Goedel_Form_Minimal_Truth_Soundness_HBL1iff_Compl_Pf* =
Goedel_Form

var trm fmla Var num
FvarsT substT Fvars subst
eql cnj imp all exi
fls
prv bprv
enc
S
P

+
Minimal_Truth_Soundness_HBL1iff_Compl_Pf

var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi
fls
dsj
num
enc
prv bprv
P
isTrue
Pf

for

var :: 'var set **and** *trm* :: 'trm set **and** *fmla* :: 'fmla set

and *Var FvarsT substT Fvars subst*

and *eql cnj imp all exi*

and *fls*

and *dsj*

and *num*

and *prv bprv*

and *enc* (⟨_⟩)

and *S*

and *isTrue*

and *P*

and *Pf*

locale *Goedel_Form_Minimal_Truth_Soundness_HBL1iff_Compl_Pf_Compl_NegPf* =

Goedel_Form_Minimal_Truth_Soundness_HBL1iff_Compl_Pf

var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi
fls
dsj
num
prv bprv
enc
S
isTrue
P

```

    Pf
+
Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  enc
  prv bprv
  P
  isTrue
  Pf
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and enc (⟨_⟩)
and S
and isTrue
and P
and Pf
+
assumes prv_ωconsistent: ωconsistent

sublocale
  Goedel_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf <
  recover_proofs: Goedel_Form_Proofs_Minimal_Truth
  where prfOf = prfOf and proof = proof and encPf = encPf
  and prv = prv and bprv = bprv
  ⟨proof⟩

context Goedel_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf begin
thm recover_proofs.goedel_first_strong

end

```

8.3 Classical-Logic Versions of Gödel's First

8.3.1 First classical-logic version

```

locale Goedel_Form_Classical_HBL1_rev_prv =
  Goedel_Form
  var trm fmla Var num FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  prv bprv
  enc
  S
  P
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set

```

and *Var num FvarsT substT Fvars subst*
and *eql cnj imp all exi*
and *fls*
and *prv bprv*
and *enc (⟨_⟩)*
and *S*
and *P*

+

assumes

— NB: we don't really need to assume classical reasoning (double negation) for all formulas, but only for the provability predicate:

classical_P_prv: $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{let } PP = (\lambda t. \text{subst } P \text{ t } xx) \text{ in}$
 $\text{prv } (\text{neg } (\text{neg } (PP \langle \varphi \rangle))) \implies \text{prv } (PP \langle \varphi \rangle)$

and

HBL1_rev_prv: $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } (PP \langle \varphi \rangle) \implies \text{prv } \varphi$

begin

lemma *HBL1_rev*:

assumes *f*: $\varphi \in \text{fmla}$ **and** *fv*: $\text{Fvars } \varphi = \{\}$ **and** *bp*: $\text{bprv } (PP \langle \varphi \rangle)$

shows $\text{prv } \varphi$

⟨proof⟩

lemma *classical_PP_prv*: $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } (\text{neg } (\text{neg } (PP \langle \varphi \rangle))) \implies \text{prv } (PP \langle \varphi \rangle)$

⟨proof⟩

lemma *HBL1_iff*: $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{bprv } (PP \langle \varphi \rangle) \longleftrightarrow \text{prv } \varphi$

⟨proof⟩

lemma *HBL1_iff_prv*: $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } (PP \langle \varphi \rangle) \longleftrightarrow \text{prv } \varphi$

⟨proof⟩

lemma *goedel_first_theHardHalf_pos*:

assumes $\text{prv } (\text{neg } \varphi G)$ **shows** $\text{prv } \text{fls}$

⟨proof⟩

corollary *goedel_first_theHardHalf*:

$\text{consistent} \implies \neg \text{prv } (\text{neg } \varphi G)$

⟨proof⟩

theorem *goedel_first_classic*:

assumes consistent

shows $\neg \text{prv } \varphi G \wedge \neg \text{prv } (\text{neg } \varphi G)$

⟨proof⟩

theorem *goedel_first_classic_ex*:

assumes consistent

shows $\exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi)$

⟨proof⟩

end — context *Goedel_Form_Classical_HBL1_rev_prv*

8.3.2 Second classical-logic version

locale *Goedel_Form_Classical_HBL1_rev_prv_Minimal_Truth_Soundness_TIP* =

Goedel_Form_Classical_HBL1_rev_prv

var trm fmla Var num FvarsT substT Fvars subst

eql cnj imp all exi

fls


```

    prv bprv
  enc
  S
  P
+
Minimal_Truth_Soundness
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  bprv
  isTrue
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var num FvarsT substT Fvars subst
and eql cnj dsj imp all exi
and fls
and prv bprv
and enc (⟦_⟧)
and S
and P
and isTrue
+
assumes
— Truth of  $\varphi$  implies provability (TIP) of (the internal representation of)  $\varphi$ 
TIP:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies$ 
  let PP = ( $\lambda t. \text{subst } P \ t \ xx$ ) in
  isTrue (PP  $\langle \varphi \rangle$ )  $\implies$  prv  $\varphi$ 
begin

lemma TIP_PP:  $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{isTrue } (PP \langle \varphi \rangle) \implies \text{prv } \varphi$ 
  ⟨proof⟩

theorem isTrue_φG:
  assumes consistent
  shows isTrue φG
  ⟨proof⟩

theorem goedel_first_classic_strong: consistent  $\implies \neg \text{prv } \varphi G \wedge \neg \text{prv } (\text{neg } \varphi G) \wedge \text{isTrue } \varphi G$ 
  ⟨proof⟩

theorem goedel_first_classic_strong_ex:
  consistent  $\implies \exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi) \wedge \text{isTrue } \varphi$ 
  ⟨proof⟩

end — context Goedel_Form_Classical_HBL1_rev_prv_Minimal_Truth_Soundness_TIP

```

8.3.3 Third classical-logic version

```

locale Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical =
  Goedel_Form
  var trm fmla Var num FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  prv bprv
  enc
  S

```

```

  P
+
Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  enc
  prv bprv
  P
  isTrue
  Pf
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and enc (⟦_⟧)
and S
and isTrue
and P
and Pf

sublocale Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical <
  recover_proofs: Goedel_Form_Classical_HBL1_rev_prv_Minimal_Truth_Soundness_TIP where prv
= prv and bprv = bprv
⟨proof⟩

context Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical begin
thm recover_proofs.goedel_first_classic_strong
end — context Goedel_Form_Minimal_Truth_Soundness_HBL1iff_prv_Compl_Pf_Classical

```

Chapter 9

Rosser Formulas

The Rosser formula is a modification of the Gödel formula that is undecidable assuming consistency only (not ω -consistency).

```
locale Rosser_Form =
  Deduct2_with_PseudoOrder
    var trm fmla Var FvarsT substT Fvars subst
    eql cnj imp all exi
    fls
    dsj
    num
    prv bprv
    Lq
    +
  Repr_Neg
    var trm fmla Var FvarsT substT Fvars subst
    eql cnj imp all exi
    fls
    num
    prv bprv
    enc
    N
    +
  Repr_SelfSubst
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
    enc
    S
    +
  HBL1
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
    enc
    P
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and eql cnj imp all exi
```

```

and fls
and prv bprv
and Lq
and dsj
and enc ( $\langle \_ \rangle$ )
and N S P

locale Rosser_Form_Proofs =
Deduct2_with_PseudoOrder
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv bprv
  Lq
  +
  Repr_Neg
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  num
  prv bprv
  enc
  N
  +
  Repr_SelfSubst
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv bprv
  enc
  S
  +
  CleanRepr_Proofs
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv bprv
  enc
  fls
  dsj
  proof prfOf
  encPf
  Pf
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
  and Var FvarsT substT Fvars subst
  and num
  and eql cnj imp all exi
  and fls
  and prv bprv
  and Lq
  and dsj and proof :: 'proof set and prfOf
  and enc ( $\langle \_ \rangle$ )
  and N
  and S

```

and $encPf Pf$

context $Rosser_Form_Proofs$
begin

definition R **where** $R = all\ zz\ (imp\ (LLq\ (Var\ zz)\ (Var\ yy))\ (all\ xx'\ (imp\ (NN\ (Var\ xx)\ (Var\ xx'))\ (neg\ (PPf\ (Var\ zz)\ (Var\ xx'))))))$

definition RR **where** $RR\ t1\ t2 = psubst\ R\ [(t1,yy), (t2,xx)]$

lemma $R[simp,intro!]$: $R \in fmla$ $\langle proof \rangle$

lemma RR_def2 :
 $t1 \in trm \implies t2 \in trm \implies xx \notin FvarsT\ t1 \implies RR\ t1\ t2 = subst\ (subst\ R\ t1\ yy)\ t2\ xx$
 $\langle proof \rangle$

definition P' **where** $P' = exi\ yy\ (cnj\ (PPf\ (Var\ yy)\ (Var\ xx))\ (RR\ (Var\ yy)\ (Var\ xx)))$

definition PP' **where** $PP'\ t = subst\ P'\ t\ xx$

lemma $Fvars_R[simp]$: $Fvars\ R = \{xx,yy\}$ $\langle proof \rangle$

lemma $[simp]$: $Fvars\ (RR\ (Var\ yy)\ (Var\ xx)) = \{yy,xx\}$ $\langle proof \rangle$

lemma $P'[simp,intro!]$: $P' \in fmla$ $\langle proof \rangle$

lemma $Fvars_P'[simp]$: $Fvars\ P' = \{xx\}$ $\langle proof \rangle$

lemma $PP'[simp,intro!]$: $t \in trm \implies PP'\ t \in fmla$
 $\langle proof \rangle$

lemma $RR[simp,intro!]$: $t1 \in trm \implies t2 \in trm \implies RR\ t1\ t2 \in fmla$
 $\langle proof \rangle$

lemma $RR_simps[simp]$:
 $n \in num \implies subst\ (RR\ (Var\ yy)\ (Var\ xx))\ n\ xx = RR\ (Var\ yy)\ n$
 $m \in num \implies n \in num \implies subst\ (RR\ (Var\ yy)\ m)\ n\ yy = RR\ n\ m$
 $\langle proof \rangle$

The Rosser modification of the Gödel formula

definition $\varphi R :: 'fmla$ **where** $\varphi R \equiv diag\ (neg\ P')$

lemma $\varphi R[simp,intro!]$: $\varphi R \in fmla$ **and** $Fvars_varR[simp]$: $Fvars\ \varphi R = \{\}$
 $\langle proof \rangle$

lemma $bprv_varR_eqv$:
 $bprv\ (eqv\ \varphi R\ (neg\ (PP'\ \langle \varphi R \rangle)))$
 $\langle proof \rangle$

lemma $bprv_imp_varR$:
 $bprv\ (imp\ (neg\ (PP'\ \langle \varphi R \rangle))\ \varphi R)$
 $\langle proof \rangle$

lemma prv_varR_eqv :
 $prv\ (eqv\ \varphi R\ (neg\ (PP'\ \langle \varphi R \rangle)))$
 $\langle proof \rangle$

lemma *prv_imp_φR*:
 prv (imp (neg (PP' ⟨φR⟩)) φR)
 ⟨*proof*⟩

end — context *Rosser_Form*

sublocale *Rosser_Form_Proofs* < *Rosser_Form* **where** $P = P$
 ⟨*proof*⟩

sublocale *Rosser_Form_Proofs* < *Goedel_Form* **where** $P = P$
 ⟨*proof*⟩

Chapter 10

Abstract Formulations of Gödel-Rosser's First Incompleteness Theorem

The development here is very similar to that of Gödel First Incompleteness Theorem. It lacks classical logical variants, since for them Rosser's trick does bring any extra value.

10.1 Proof-Theoretic Versions

context *Rosser_Form_Proofs*
begin

lemma *NN_neg_unique_xx'*:
 assumes [*simp*]: $\varphi \in \text{fmla } F\text{vars } \varphi = \{\}$
 shows
 $\text{bprv } (\text{imp } (NN \langle \varphi \rangle (Var \text{ } xx'))$
 $(\text{eql } \langle \text{neg } \varphi \rangle (Var \text{ } xx'))$
 $\langle \text{proof} \rangle$

lemma *NN_imp_xx'*:
 assumes [*simp*]: $\varphi \in \text{fmla } F\text{vars } \varphi = \{\}$ $\chi \in \text{fmla}$
 shows $\text{bprv } (\text{imp } (\text{subst } \chi \langle \text{neg } \varphi \rangle xx')$
 $(\text{all } xx' (\text{imp } (NN \langle \varphi \rangle (Var \text{ } xx')) \chi)))$
 $\langle \text{proof} \rangle$

lemma *goedel_rosser_first_theEasyHalf*:
 assumes *c*: *consistent*
 shows $\neg \text{prv } \varphi R$
 $\langle \text{proof} \rangle$

lemma *goedel_rosser_first_theHardHalf*:
 assumes *c*: *consistent*
 shows $\neg \text{prv } (\text{neg } \varphi R)$
 $\langle \text{proof} \rangle$

theorem *goedel_rosser_first*:
 assumes *consistent*
 shows $\neg \text{prv } \varphi R \wedge \neg \text{prv } (\text{neg } \varphi R)$
 $\langle \text{proof} \rangle$

```

theorem goedel_rosser_first_ex:
  assumes consistent
  shows  $\exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi)$ 
   $\langle \text{proof} \rangle$ 

```

end — context *Rosser_Form*

10.2 Model-Theoretic Versions

10.2.1 First model-theoretic version

```

locale Rosser_Form_Proofs_Minimal_Truth =
  Rosser_Form_Proofs
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  fls
  prv bprv
  Lq
  dsj
  proof prfOf
  enc
  N S
  encPf
  Pf
+
  Minimal_Truth_Soundness
  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  bprv
  isTrue
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and Lq
and prv bprv
and enc ( $\langle \_ \rangle$ )
and N S P
and proof :: 'proof set and prfOf encPf
and Pf
and isTrue
begin

```

```

lemma Fvars_PP'[simp]: Fvars (PP'  $\langle \varphi R \rangle$ ) = {}  $\langle \text{proof} \rangle$ 

```

```

lemma Fvars_RR'[simp]: Fvars (RR (Var yy)  $\langle \varphi R \rangle$ ) = {yy}
   $\langle \text{proof} \rangle$ 

```

```

lemma isTrue_PPf_implies_φR:
assumes isTrue (all yy (neg (PPf (Var yy)  $\langle \varphi R \rangle$ ))))
(is isTrue ?H)

```


shows *isTrue* φR
 ⟨*proof*⟩

theorem *isTrue_φR*:
assumes *consistent*
shows *isTrue* φR
 ⟨*proof*⟩

theorem *goedel_rosser_first_strong*: *consistent* $\implies \neg \text{prv } \varphi R \wedge \neg \text{prv } (\text{neg } \varphi R) \wedge \text{isTrue } \varphi R$
 ⟨*proof*⟩

theorem *goedel_rosser_first_strong_ex*:
consistent $\implies \exists \varphi. \varphi \in \text{fmla} \wedge \neg \text{prv } \varphi \wedge \neg \text{prv } (\text{neg } \varphi) \wedge \text{isTrue } \varphi$
 ⟨*proof*⟩

end — context *Rosser_Form_Proofs_Minimal_Truth*

10.2.2 Second model-theoretic version

context *Rosser_Form*
begin
print_context
end

locale *Rosser_Form_Minimal_Truth_Soundness_HBL1iff_Compl_Pf* =
Rosser_Form
var trm fmla Var
FvarsT substT Fvars subst
num
eql cnj imp all exi
fls
prv bprv
Lq
dsj
enc
N
S
P
 +
Minimal_Truth_Soundness_HBL1iff_Compl_Pf
var trm fmla Var FvarsT substT Fvars subst
eql cnj imp all exi
fls
dsj
num
enc
prv bprv
P
isTrue
Pf
for
var :: '*var set* **and** *trm* :: '*trm set* **and** *fmla* :: '*fmla set*
and *Var FvarsT substT Fvars subst*
and *eql cnj imp all exi*
and *fls*
and *dsj*
and *num*

```

and prv bprv
and Lq
and enc (⟨_⟩)
and N S
and isTrue
and P Pf

```

```

locale Rosser_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf =
Rosser_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf

```

```

  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv bprv
  Lq
  enc
  N S
  isTrue
  P
  Pf

```

```
+
```

```

M : Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf

```

```

  var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  enc
  prv bprv
  N
  isTrue
  Pf

```

```
for
```

```

var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv bprv
and Lq
and enc (⟨_⟩)
and N S P
and isTrue
and Pf

```

```
sublocale
```

```

Rosser_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf <
recover_proofs: Rosser_Form_Proofs_Minimal_Truth
where prfOf = prfOf and proof = proof and encPf = encPf
and prv = prv and bprv = bprv
⟨proof⟩

```

```

context Rosser_Form_Minimal_Truth_Soundness_HBL1iff_Cmpl_Pf_Cmpl_NegPf
begin

```

thm *recover_proofs.goedel_rosser_first_strong*
end

Chapter 11

Abstract Formulation of Gödel's Second Incompleteness Theorem

We assume all three derivability conditions, and assumptions behind Gödel formulas:

```
locale Goedel_Second_Assumptions =  
HBL1_2_3  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
  eql cnj imp all exi  
  prv bprv  
  enc  
  P  
+  
Goedel_Form  
  var trm fmla Var num FvarsT substT Fvars subst  
  eql cnj imp all exi  
  fls  
  prv bprv  
  enc  
  S  
  P  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and num  
and eql cnj imp all exi  
and prv bprv  
and enc ((\_))  
and S  
and P  
and fls  
begin  
  
lemma P_G:  
bprv (imp (PP <φG>) (PP <fls>))  
<proof>
```

First the "direct", positive formulation:

```
lemma goedel_second_pos:  
assumes prv (neg (PP <fls>))  
shows prv fls  
<proof>
```

Then the more standard, counterpositive formulation:

theorem *goedel_second*:

consistent $\implies \neg \text{prv} (\text{neg} (PP \langle \text{fls} \rangle))$

<proof>

It is an immediate consequence of Gödel's Second HLB1, HLB2 that (assuming consistency) *prv* (*neg* (*PP* $\langle \varphi \rangle$)) holds for no sentence, be it provable or not. The theory is omniscient about what it can prove (thanks to HLB1), but completely ignorant about what it cannot prove.

corollary *not_prv_neg_PP*:

assumes *c*: *consistent* **and** [*simp*]: $\varphi \in \text{fmla } F\text{vars } \varphi = \{\}$

shows $\neg \text{prv} (\text{neg} (PP \langle \varphi \rangle))$

<proof>

end — context *Goedel_Second_Assumptions*

Chapter 12

Jeroslow's Variant of Gödel's Second Incompleteness Theorem

12.1 Encodings and Derivability

Here we formalize some of the assumptions of Jeroslow's theorem: encoding, term-encoding and the First Derivability Condition.

12.1.1 Encoding of formulas

```
locale Encode =  
  Syntax_with_Numerals  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var FvarsT substT Fvars subst  
and num  
+  
fixes
```

```
enc :: 'fmla  $\Rightarrow$  'trm (( $\_$ ))  
assumes  
enc[simp,intro!]:  $\bigwedge \varphi. \varphi \in fmla \implies enc \varphi \in num$   
begin
```

```
end — context Encode
```

12.1.2 Encoding of computable functions

Jeroslow assumes the encodability of an abstract (unspecified) class of computable functions and the assumption that a particular function, *sub* φ for each formula φ , is in this collection. This is used to prove a different flavor of the diagonalization lemma (Jeroslow 1973). It turns out that only an encoding of unary computable functions is needed, so we only assume that.

```
locale Encode_UComput =  
  Encode  
  var trm fmla Var FvarsT substT Fvars subst  
  num  
  enc  
for
```

```

var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and enc (<_>)
+
— Abstract (unspecified) notion of unary "computable" function between numerals, which are encoded as
numerals. They contain a special substitution-like function sub  $\varphi$  for each formula  $\varphi$ .
fixes ufunc :: ('trm  $\Rightarrow$  'trm) set
and encF :: ('trm  $\Rightarrow$  'trm)  $\Rightarrow$  'trm
and sub :: 'fmla  $\Rightarrow$  'trm  $\Rightarrow$  'trm
assumes
— NB: Due to the limitations of the type system, we define ufunc as a set of functions between terms,
but we only care about their actions on numerals ... so we assume they send numerals to numerals:
ufunc[simp,intro!]:  $\bigwedge f n. f \in \text{ufunc} \implies n \in \text{num} \implies f n \in \text{num}$ 
and
encF[simp,intro!]:  $\bigwedge f. f \in \text{ufunc} \implies \text{encF } f \in \text{num}$ 
and
sub[simp]:  $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{sub } \varphi \in \text{ufunc}$ 
and
— The function sub  $\varphi$  takes any encoding of a function f and returns the encoding of the formula obtained
by substituting for xx the value of f applied to its own encoding:
sub_enc:
 $\bigwedge \varphi f. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{xx\} \implies f \in \text{ufunc} \implies$ 
 $\text{sub } \varphi (\text{encF } f) = \text{enc } (\text{inst } \varphi (f (\text{encF } f)))$ 

```

12.1.3 Term-encoding of computable functions

For handling the notion of term-representation (which we introduce later), we assume we are given a set *Ops* of term operators and their encodings as numerals. We additionally assume that the term operators behave well w.r.t. free variables and substitution.

```

locale TermEncode =
Syntax_with_Numerals
var trm fmla Var FvarsT substT Fvars subst
num
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
+
fixes
Ops :: ('trm  $\Rightarrow$  'trm) set
and
enc :: ('trm  $\Rightarrow$  'trm)  $\Rightarrow$  'trm (<_>)
assumes
Ops[simp,intro!]:  $\bigwedge f t. f \in \text{Ops} \implies t \in \text{trm} \implies f t \in \text{trm}$ 
and
enc[simp,intro!]:  $\bigwedge f. f \in \text{Ops} \implies \text{enc } f \in \text{num}$ 
and
Ops_FvarsT[simp]:  $\bigwedge f t. f \in \text{Ops} \implies t \in \text{trm} \implies \text{FvarsT } (f t) = \text{FvarsT } t$ 
and
Ops_substT[simp]:  $\bigwedge f t. f \in \text{Ops} \implies t \in \text{trm} \implies t1 \in \text{trm} \implies x \in \text{var} \implies$ 
 $\text{substT } (f t) t1 x = f (\text{substT } t t1 x)$ 
begin
end — context TermEncode

```

12.1.4 The first Hilbert-Bernays-Löb derivability condition

```

locale HBL1 =
  Encode
    var trm fmla Var FvarsT substT Fvars subst
    num
    enc
  +
  Deduct
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and num
and eql cnj imp all exi
and prv bprv
and enc (⟨_⟩)
  +
fixes P :: 'fmla
assumes
  P[intro!,simp]: P ∈ fmla
and
  Fvars_P[simp]: Fvars P = {xx}
and
  HBL1:  $\bigwedge \varphi. \varphi \in fmla \implies Fvars \varphi = \{\} \implies prv \varphi \implies prv (subst P \langle \varphi \rangle xx)$ 
begin

```

Predicate version of the provability formula

definition PP **where** PP $\equiv \lambda t. subst P t xx$

lemma PP[*simp*]: $\bigwedge t. t \in trm \implies PP t \in fmla$
 ⟨*proof*⟩

lemma Fvars_PP[*simp*]: $\bigwedge t. t \in trm \implies Fvars (PP t) = FvarsT t$
 ⟨*proof*⟩

lemma [*simp*]:
 $n \in num \implies subst (PP (Var yy)) n yy = PP n$
 $n \in num \implies subst (PP (Var xx)) n xx = PP n$
 ⟨*proof*⟩

lemma HBL1_PP:
 $\varphi \in fmla \implies Fvars \varphi = \{\} \implies prv \varphi \implies prv (PP \langle \varphi \rangle)$
 ⟨*proof*⟩

end — context HBL1

12.2 A Formalization of Jeroslow's Original Argument

12.2.1 Preliminaries

The First Derivability Condition was stated using a formula with free variable xx , whereas the pseudo-term theory employs a different variable, inp . The distinction is of course immaterial, because we can perform a change of variable in the instantiation:

context *HBL1*
begin

Changing the variable (from *xx* to *inp*) in the provability predicate:

definition *Pinp* \equiv *subst P (Var inp) xx*
lemma *PP_Pinp*: $t \in \text{trm} \implies PP\ t = \text{instInp Pinp } t$
<proof>

A version of HBL1 that uses the *inp* variable:

lemma *HBL1_inp*:
 $\varphi \in \text{fmLa} \implies \text{Fvars } \varphi = \{\} \implies \text{prv } \varphi \implies \text{prv } (\text{instInp Pinp } \langle \varphi \rangle)$
<proof>

end — context *HBL1*

12.2.2 Jeroslow-style diagonalization

locale *Jeroslow_Diagonalization* =
Deduct_with_False_Disj_Rename
var trm fmLa Var FvarsT substT Fvars subst
eql conj imp all exi
fls
dsj
num
prv
 +
Encode
var trm fmLa Var FvarsT substT Fvars subst
num
enc
for
var :: '*var set and trm* :: '*trm set and fmLa* :: '*fmLa set*
and *Var FvarsT substT Fvars subst*
and *eql conj imp all exi*
and *fls*
and *dsj*
and *num*
and *prv*
and *enc* (*<_>*)
 +
fixes *F* :: ('*trm* \Rightarrow '*trm*) *set*
and *encF* :: ('*trm* \Rightarrow '*trm*) \Rightarrow '*fmLa*
and *N* :: '*trm* \Rightarrow '*trm*
and *ssap* :: '*fmLa* \Rightarrow '*trm* \Rightarrow '*trm*
assumes
 — For the members *f* of *F*, we will only care about their action on numerals, and we assume that they send numerals to numerals.
F[simp,intro!]: $\bigwedge f\ n. f \in F \implies n \in \text{num} \implies f\ n \in \text{num}$
and
encF[simp,intro!]: $\bigwedge f. f \in F \implies \text{encF } f \in \text{ptrm } (\text{Suc } 0)$
and
N[simp,intro!]: $N \in F$
and
ssap[simp]: $\bigwedge \varphi. \varphi \in \text{fmLa} \implies \text{Fvars } \varphi = \{\text{inp}\} \implies \text{ssap } \varphi \in F$
and
ReprF: $\bigwedge f\ n. f \in F \implies n \in \text{num} \implies \text{prveqlPT } (\text{instInp } (\text{encF } f)\ n)\ (f\ n)$
and
CapN: $\bigwedge \varphi. \varphi \in \text{fmLa} \implies \text{Fvars } \varphi = \{\} \implies N\ \langle \varphi \rangle = \langle \text{neg } \varphi \rangle$

and

CapSS: — We consider formulas ψ of one variable, called *inp*:

$\bigwedge \psi f. \psi \in fmla \implies Fvars \psi = \{inp\} \implies f \in F \implies$

$ssap \psi \langle encF f \rangle = \langle instInpP \psi 0 (instInp (encF f) \langle encF f \rangle) \rangle$

begin

lemma *encF_fm1a[simp,intro!]*: $\bigwedge f. f \in F \implies encF f \in fmla$

<proof>

lemma *enc_trm*: $\varphi \in fmla \implies \langle \varphi \rangle \in trm$

<proof>

definition $\tau J :: 'fmla \Rightarrow 'fmla$ **where**

$\tau J \psi \equiv instInp (encF (ssap \psi)) (\langle encF (ssap \psi) \rangle)$

definition $\varphi J :: 'fmla \Rightarrow 'fmla$ **where**

$\varphi J \psi \equiv instInpP \psi 0 (\tau J \psi)$

lemma *τJ [simp]*:

assumes $\psi \in fmla$ **and** $Fvars \psi = \{inp\}$

shows $\tau J \psi \in ptrm 0$

<proof>

lemma *τJ _fm1a[simp]*:

assumes $\psi \in fmla$ **and** $Fvars \psi = \{inp\}$

shows $\tau J \psi \in fmla$

<proof>

lemma *FvarsT_ τJ [simp]*:

assumes $\psi \in fmla$ **and** $Fvars \psi = \{inp\}$

shows $Fvars (\tau J \psi) = \{out\}$

<proof>

lemma *φJ [simp]*:

assumes $\psi \in fmla$ **and** $Fvars \psi = \{inp\}$

shows $\varphi J \psi \in fmla$

<proof>

lemma *Fvars_ φJ [simp]*:

assumes $\psi \in fmla$ **and** $Fvars \psi = \{inp\}$

shows $Fvars (\varphi J \psi) = \{\}$

<proof>

lemma *diagonalization*:

assumes $\psi[simp]: \psi \in fmla$ **and** $[simp]: Fvars \psi = \{inp\}$

shows $prveq1PT (\tau J \psi) \langle instInpP \psi 0 (\tau J \psi) \rangle \wedge$

$prv (eqv (\varphi J \psi) (instInp \psi \langle \varphi J \psi \rangle))$

<proof>

end — context *Jeroslow_Diagonalization*

12.2.3 Jeroslow's Second Incompleteness Theorem

We follow Jeroslow's pseudo-term-based development of the Second Incompleteness Theorem and point out the location in the proof that implicitly uses an unstated assumption: the fact that, for certain two provably equivalent formulas φ and φ' , it is provable that the provability of the encoding of φ' implies the provability of the encoding of φ .

locale *Jeroslow_Godel_Second* =

Jeroslow_Diagonalization

```

var trm fmla Var FvarsT substT Fvars subst
  eql cnj imp all exi
  fls
  dsj
  num
  prv
  enc
  F encF N ssap
+

```

HBL1

```

var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv prv
  enc
  P

```

for

```

var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and dsj
and num
and prv
and enc (<_>)
and P
and F encF N ssap
+

```

assumes

SHBL3: $\bigwedge \tau. \tau \in ptrm\ 0 \implies prv\ (imp\ (instInpP\ Pinp\ 0\ \tau)\ (instInp\ Pinp\ \langle instInpP\ Pinp\ 0\ \tau \rangle))$

begin

Consistency formula a la Jeroslow:

definition *jcons* :: 'fmla **where**

```

jcons  $\equiv all\ xx\ (neg\ (cnj\ (instInp\ Pinp\ (Var\ xx))\ (instInpP\ Pinp\ 0\ (instInp\ (encF\ N)\ (Var\ (xx))))))$ 

```

lemma *prv_eql_subst_trm3*:

```

x  $\in var \implies \varphi \in fmla \implies t1 \in trm \implies t2 \in trm \implies$ 
prv  $(eql\ t1\ t2) \implies prv\ (subst\ \varphi\ t1\ x) \implies prv\ (subst\ \varphi\ t2\ x)$ 
<proof>

```

lemma *Pinp[simp,intro!]*: *Pinp* \in fmla

```

and Fvars_Pinp[simp]: Fvars Pinp = {inp}
<proof>

```

lemma *ReprF_combineWith_CapN*:

```

assumes  $\varphi \in fmla$  and Fvars  $\varphi = \{\}$ 
shows prveqlPT  $(instInp\ (encF\ N)\ \langle \varphi \rangle)\ \langle neg\ \varphi \rangle$ 
<proof>

```

theorem *jeroslow_godel_second*:

assumes *consistent*

— Assumption that is not stated by Jeroslow, but seems to be needed:

assumes *unstated*:

```

  let  $\psi = instInpP\ Pinp\ (Suc\ 0)\ (encF\ N);$ 
     $\tau = \tau J\ \psi;$ 

```

```

     $\varphi = \text{instInpP } (\text{instInpP } \text{Pinp } (\text{Suc } 0) (\text{encF } N)) 0 \ \tau;$ 
     $\varphi' = \text{instInpP } \text{Pinp } 0 (\text{instInpP } (\text{encF } N) 0 \ \tau)$ 
    in prv (imp (instInp Pinp  $\langle\varphi'\rangle$ ) (instInp Pinp  $\langle\varphi\rangle$ ))
shows  $\neg$  prv jcons
  <proof>

end — context Jeroslow_Godel_Second

```

12.3 A Simplification of Jeroslow's Original Argument

This is the simplified version of Jeroslow's Second Incompleteness Theorem reported in our CADE 2019 paper [1]. The simplification consists of replacing pseudo-terms with plain terms and representability with (what we call in the paper) term-representability. This simplified version does not incur the complications of the original.

12.3.1 Jeroslow-style term-based diagonalization

```

locale Jeroslow_Diagonalization =
  Deduct_with_False
    var trm fmla Var FvarsT substT Fvars subst
    eql cnj imp all exi
    fls
    num
    prv
  +
  Encode
    var trm fmla Var FvarsT substT Fvars subst
    num
    enc
  +
  TermEncode
    var trm fmla Var FvarsT substT Fvars subst
    num
    Ops tenc
for
  var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var FvarsT substT Fvars subst
and eql cnj imp all exi
and fls
and num
and prv
and enc ( $\langle\_ \rangle$ )
and Ops and tenc
  +
fixes F :: ('trm  $\Rightarrow$  'trm) set
  and encF :: ('trm  $\Rightarrow$  'trm)  $\Rightarrow$  ('trm  $\Rightarrow$  'trm)
  and N :: 'trm  $\Rightarrow$  'trm
  and ssap :: 'fmla  $\Rightarrow$  'trm  $\Rightarrow$  'trm
assumes
  F[simp,intro!]:  $\bigwedge f n. f \in F \Longrightarrow n \in \text{num} \Longrightarrow f n \in \text{num}$ 
and
  encF[simp,intro!]:  $\bigwedge f. f \in F \Longrightarrow \text{encF } f \in \text{Ops}$ 
and
  N[simp,intro!]:  $N \in F$ 
and
  ssap[simp]:  $\bigwedge \varphi. \varphi \in \text{fmla} \Longrightarrow \text{Fvars } \varphi = \{xx\} \Longrightarrow \text{ssap } \varphi \in F$ 

```

and
ReprF: $\bigwedge f n. f \in F \implies n \in \text{num} \implies \text{prv} (\text{eql} (\text{encF } f \ n) (f \ n))$
and
CapN: $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies N \langle \varphi \rangle = \langle \text{neg } \varphi \rangle$
and
CapSS:
 $\bigwedge \psi f. \psi \in \text{fmla} \implies \text{Fvars } \psi = \{xx\} \implies f \in F \implies$
 $\text{ssap } \psi (\text{tenc} (\text{encF } f)) = \langle \text{inst } \psi (\text{encF } f (\text{tenc} (\text{encF } f))) \rangle$
begin

definition *tJ* :: 'fmla \Rightarrow 'trm **where**
tJ $\psi \equiv \text{encF} (\text{ssap } \psi) (\text{tenc} (\text{encF} (\text{ssap } \psi)))$

definition φJ :: 'fmla \Rightarrow 'fmla **where**
 $\varphi J \psi \equiv \text{subst } \psi (tJ \psi) \ xx$

lemma *tJ[simp]*:
assumes $\psi \in \text{fmla}$ **and** $\text{Fvars } \psi = \{xx\}$
shows $tJ \psi \in \text{trm}$
<proof>

lemma *FvarsT_tJ[simp]*:
assumes $\psi \in \text{fmla}$ **and** $\text{Fvars } \psi = \{xx\}$
shows $\text{FvarsT} (tJ \psi) = \{\}$
<proof>

lemma φJ [*simp*]:
assumes $\psi \in \text{fmla}$ **and** $\text{Fvars } \psi = \{xx\}$
shows $\varphi J \psi \in \text{fmla}$
<proof>

lemma *Fvars_φJ[simp]*:
assumes $\psi \in \text{fmla}$ **and** $\text{Fvars } \psi = \{xx\}$
shows $\text{Fvars} (\varphi J \psi) = \{\}$
<proof>

lemma *diagonalization*:
assumes $\psi \in \text{fmla}$ **and** $\text{Fvars } \psi = \{xx\}$
shows $\text{prv} (\text{eql} (tJ \psi) (\text{inst } \psi (tJ \psi))) \wedge$
 $\text{prv} (\text{eqv} (\varphi J \psi) (\text{inst } \psi (\varphi J \psi)))$
<proof>

end — context *Jeroslow_Diagonalization*

12.3.2 Term-based version of Jeroslow's Second Incompleteness Theorem

locale *Jeroslow_Godel_Second* =
Jeroslow_Diagonalization
var *trm* *fmla* *Var* *FvarsT* *substT* *Fvars* *subst*
eql *cnj* *imp* *all* *exi*
fls
num
prv
enc
Ops *tenc*
F *encF* *N* *ssap*
+

HBL1
var trm fmla Var FvarsT substT Fvars subst
num
eql cnj imp all exi
prv prv
enc
P
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and *Var FvarsT substT Fvars subst*
and *eql cnj imp all exi*
and *fls*
and *num*
and *prv*
and *enc (⟨_⟩)*
and *Ops and tenc*
and *P*
and *F encF N ssap*
 +
assumes
SHBL3: $\bigwedge t. t \in \text{trm} \implies \text{FvarsT } t = \{\} \implies \text{prv } (\text{imp } (PP \ t) \ (PP \ \langle PP \ t \rangle))$
begin

Consistency formula a la Jeroslow:

definition *jcons :: 'fmla where*
jcons $\equiv \text{all } xx \ (\text{neg } (\text{cnj } (PP \ (\text{Var } xx)) \ (PP \ (\text{encF } N \ (\text{Var } (xx))))))$

lemma *prv_eql_subst_trm3:*
 $x \in \text{var} \implies \varphi \in \text{fmla} \implies t1 \in \text{trm} \implies t2 \in \text{trm} \implies$
 $\text{prv } (\text{eql } t1 \ t2) \implies \text{prv } (\text{subst } \varphi \ t1 \ x) \implies \text{prv } (\text{subst } \varphi \ t2 \ x)$
⟨proof⟩

lemma *prv_eql_neg_encF_N:*
assumes *$\varphi \in \text{fmla}$ and $\text{Fvars } \varphi = \{\}$*
shows *$\text{prv } (\text{eql } \langle \text{neg } \varphi \rangle \ (\text{encF } N \ \langle \varphi \rangle))$*
⟨proof⟩

lemma *prv_imp_neg_encF_N_aux:*
assumes *$\varphi \in \text{fmla}$ and $\text{Fvars } \varphi = \{\}$*
shows *$\text{prv } (\text{imp } (PP \ \langle \text{neg } \varphi \rangle) \ (PP \ (\text{encF } N \ \langle \varphi \rangle)))$*
⟨proof⟩

lemma *prv_cnj_neg_encF_N_aux:*
assumes *$\varphi \in \text{fmla}$ and $\text{Fvars } \varphi = \{\}$ $\chi \in \text{fmla}$ $\text{Fvars } \chi = \{\}$*
and *$\text{prv } (\text{neg } (\text{cnj } \chi \ (PP \ \langle \text{neg } \varphi \rangle)))$*
shows *$\text{prv } (\text{neg } (\text{cnj } \chi \ (PP \ (\text{encF } N \ \langle \varphi \rangle))))$*
⟨proof⟩

theorem *jeroslow_godel_second:*
assumes *consistent*
shows *$\neg \text{prv } \text{jcons}$*
⟨proof⟩

12.3.3 A variant of the Second Incompleteness Theorem

This variant (also discussed in our CADE 2019 paper [1]) strengthens the conclusion of the theorem to the standard formulation of "does not prove its own consistency" at the expense of two additional derivability-like conditions, HBL4 and WHBL2.

theorem *jeroslow_godel_second_standardCon*:
assumes *consistent*
and *HBL4*: $\bigwedge \varphi 1 \varphi 2. \{\varphi 1, \varphi 2\} \subseteq \text{fmla} \implies \text{Fvars } \varphi 1 = \{\} \implies \text{Fvars } \varphi 2 = \{\} \implies$
 $\text{prv } (\text{imp } (\text{cnj } (\text{PP } \langle \varphi 1 \rangle) (\text{PP } \langle \varphi 2 \rangle)) (\text{PP } \langle \text{cnj } \varphi 1 \varphi 2 \rangle))$
and *WHBL2*: $\bigwedge \varphi 1 \varphi 2. \{\varphi 1, \varphi 2\} \subseteq \text{fmla} \implies \text{Fvars } \varphi 1 = \{\} \implies \text{Fvars } \varphi 2 = \{\} \implies$
 $\text{prv } (\text{imp } \varphi 1 \varphi 2) \implies \text{prv } (\text{imp } (\text{PP } \langle \varphi 1 \rangle) (\text{PP } \langle \varphi 2 \rangle))$
shows $\neg \text{prv } (\text{neg } (\text{PP } \langle \text{fls} \rangle))$
 $\langle \text{proof} \rangle$

Next we perform a formal analysis of some connection between the above theorems' hypotheses.

definition *noContr* :: *bool* **where**
 $\text{noContr} \equiv \forall \varphi \in \text{fmla}. \text{Fvars } \varphi = \{\} \longrightarrow \text{prv } (\text{neg } (\text{cnj } (\text{PP } \langle \varphi \rangle) (\text{PP } \langle \text{neg } \varphi \rangle)))$

lemma *jcons_noContr*:
assumes *j*: *prv jcons*
shows *noContr*
 $\langle \text{proof} \rangle$

noContr is still stronger than the standard notion of proving own consistency:

lemma *noContr_implies_neg_PP_fl*:
assumes *noContr*
shows $\text{prv } (\text{neg } (\text{PP } \langle \text{fls} \rangle))$
 $\langle \text{proof} \rangle$

corollary *jcons_implies_neg_PP_fl*:
assumes *prv jcons*
shows $\text{prv } (\text{neg } (\text{PP } \langle \text{fls} \rangle))$
 $\langle \text{proof} \rangle$

However, unlike *jcons*, which seems to be quite a bit stronger, *noContr* is equivalent to the standard notion under a slightly stronger assumption than our *WWHBL2*, namely, a binary version of that:

lemma *neg_PP_fl_implies_noContr*:
assumes *WWHBL2*:
 $\bigwedge \varphi \chi \psi. \varphi \in \text{fmla} \implies \chi \in \text{fmla} \implies \psi \in \text{fmla} \implies$
 $\text{Fvars } \varphi = \{\} \implies \text{Fvars } \chi = \{\} \implies \text{Fvars } \psi = \{\} \implies$
 $\text{prv } (\text{imp } \varphi (\text{imp } \chi \psi)) \implies \text{prv } (\text{imp } (\text{PP } \langle \varphi \rangle) (\text{imp } (\text{PP } \langle \chi \rangle) (\text{PP } \langle \psi \rangle)))$
assumes *p*: $\text{prv } (\text{neg } (\text{PP } \langle \text{fls} \rangle))$
shows *noContr*
 $\langle \text{proof} \rangle$

end — context *Jeroslow_Godel_Second*

Chapter 13

Löb Formulas

The Löb formula, parameterized by a sentence φ , is defined by diagonalizing $\text{imp } P \varphi$.

```
locale Loeb_Form =
  Deduct2
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
  +
  Repr_SelfSubst
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
    enc
    S
  +
  HBL1
    var trm fmla Var FvarsT substT Fvars subst
    num
    eql cnj imp all exi
    prv bprv
    enc
    P
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var num FvarsT substT Fvars subst
and eql cnj imp all exi
and prv bprv
and enc (\_)
and S
and P
begin
```

The Löb formula associated to a formula φ :

definition $\varphi L :: 'fmla \Rightarrow 'fmla$ **where** $\varphi L \varphi \equiv \text{diag } (\text{imp } P \varphi)$

lemma $\varphi L[\text{simp}, \text{intro}]$: $\bigwedge \varphi. \varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \varphi L \varphi \in \text{fmla}$

and

$\text{Fvars_}\varphi L[\text{simp}]$: $\varphi \in \text{fmla} \implies \text{Fvars } \varphi = \{\} \implies \text{Fvars } (\varphi L \varphi) = \{\}$
(*proof*)

lemma $\text{bprv_}\varphi L_ \text{eqv}$:

$\varphi \in \text{fm}la \implies \text{Fvars } \varphi = \{\} \implies \text{bprv } (\text{eqv } (\varphi L \ \varphi) \ (\text{imp } (PP \ \langle \varphi L \ \varphi \rangle) \ \varphi))$
<proof>

lemma *prv_φL_eqv:*

$\varphi \in \text{fm}la \implies \text{Fvars } \varphi = \{\} \implies \text{prv } (\text{eqv } (\varphi L \ \varphi) \ (\text{imp } (PP \ \langle \varphi L \ \varphi \rangle) \ \varphi))$
<proof>

end — context *Loeb_Form*

Chapter 14

Löb's Theorem

We have set up the formalization of Gödel's first (easy half) and Gödel's second so that the following generalizations, leading to Löb's theorem, are trivial modifications of these, replacing negation with "implies φ " in all proofs.

```
locale Loeb_Assumptions =
  HBL1_2_3
  var trm fmla Var FvarsT substT Fvars subst
  num
  eql cnj imp all exi
  prv bprv
  enc
  P
+
  Loeb_Form
  var trm fmla Var num FvarsT substT Fvars subst
  eql cnj imp all exi
  prv bprv
  enc
  S
  P
for
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set
and Var num FvarsT substT Fvars subst
and eql cnj imp all exi
and prv bprv
and enc (⟦_⟧)
and S
and P
begin
```

Generalization of *goedel_first_theEasyHalf_pos*, replacing *fls* with a sentence φ :

```
lemma loeb_aux_prv:
assumes  $\varphi[simp]$ :  $\varphi \in fmla Fvars \varphi = \{\}$  and  $p$ : prv ( $\varphi L \varphi$ )
shows prv  $\varphi$ 
⟨proof⟩
```

```
lemma loeb_aux_bprv:
assumes  $\varphi[simp]$ :  $\varphi \in fmla Fvars \varphi = \{\}$  and  $p$ : bprv ( $\varphi L \varphi$ )
shows bprv  $\varphi$ 
⟨proof⟩
```

Generalization of *P_G*, the main lemma used for Gödel's second:

lemma *P_L*:
assumes $\varphi[simp]$: $\varphi \in fmla \ Fvars \ \varphi = \{\}$
shows $bprv \ (imp \ (PP \ \langle\varphi L \ \varphi\rangle) \ (PP \ \langle\varphi\rangle))$
 $\langle proof \rangle$

Löb's theorem generalizes the positive formulation Gödel's Second (*goedel_second*). In our two-provability-relation framework, we get two variants of Löb's theorem. A stronger variant, assuming *prv* and proving *bprv*, seems impossible.

theorem *loeb_bprv*:
assumes $\varphi[simp]$: $\varphi \in fmla \ Fvars \ \varphi = \{\}$ **and** p : $bprv \ (imp \ (PP \ \langle\varphi\rangle) \ \varphi)$
shows $bprv \ \varphi$
 $\langle proof \rangle$

theorem *loeb_prv*:
assumes $\varphi[simp]$: $\varphi \in fmla \ Fvars \ \varphi = \{\}$ **and** p : $prv \ (imp \ (PP \ \langle\varphi\rangle) \ \varphi)$
shows $prv \ \varphi$
 $\langle proof \rangle$

We could have of course inferred *goedel_first_theEasyHalf_pos* and *goedel_second* from these more general versions, but we leave the original arguments as they are more instructive.

end — context *Loeb_Assumptions*

Chapter 15

Abstract Formulation of Tarski's Theorems

We prove Tarski's proof-theoretic and semantic theorems about the non-definability and respectively non-expressiveness (in the standard model) of truth

15.1 Non-Definability of Truth

context *Goedel_Form*
begin

context
 fixes $T :: 'fmla$
 assumes $T[simp,intro!]: T \in fmla$
 and $Fvars_T[simp]: Fvars\ T = \{xx\}$
 and $prv_T: \bigwedge \varphi. \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies prv\ (eqv\ (subst\ T\ \langle \varphi \rangle\ xx)\ \varphi)$
begin

definition $\varphi T :: 'fmla$ **where** $\varphi T \equiv diag\ (neg\ T)$

lemma $\varphi T[simp,intro!]: \varphi T \in fmla$ **and**
 $Fvars_ \varphi T[simp]: Fvars\ \varphi T = \{\}$
<proof>

lemma $bprv_ \varphi T_ eqv:$
 $bprv\ (eqv\ \varphi T\ (neg\ (subst\ T\ \langle \varphi T \rangle\ xx)))$
<proof>

lemma $prv_ \varphi T_ eqv:$
 $prv\ (eqv\ \varphi T\ (neg\ (subst\ T\ \langle \varphi T \rangle\ xx)))$
<proof>

lemma $\varphi T_ prv_ fls: prv\ fls$
<proof>

end — context

theorem *Tarski_proof_theoretic:*
assumes $T \in fmla\ Fvars\ T = \{xx\}$
and $\bigwedge \varphi. \varphi \in fmla \implies Fvars\ \varphi = \{\} \implies prv\ (eqv\ (subst\ T\ \langle \varphi \rangle\ xx)\ \varphi)$
shows $\neg\ consistent$

<proof>

end — context *Goedel_Form*

15.2 Non-Expressiveness of Truth

This follows as a corollary of the syntactic version, after taking *prv* to be *isTrue* on sentences. Indeed, this is a virtue of our abstract treatment of provability: We don't work with a particular predicate, but with any predicate that is closed under some rules — which could as well be a semantic notion of truth (for sentences).

```
locale Goedel_Form_prv_eq_isTrue =  
  Goedel_Form  
    var trm fmla Var num FvarsT substT Fvars subst  
    eql cnj imp all exi  
    fls  
    prv bprv  
    enc  
    P  
    S  
for  
var :: 'var set and trm :: 'trm set and fmla :: 'fmla set  
and Var num FvarsT substT Fvars subst  
and eql cnj imp all exi  
and fls  
and prv bprv  
and enc (<_>)  
and S  
and P  
+  
fixes isTrue :: 'fmla  $\Rightarrow$  bool  
assumes prv_eq_isTrue:  $\bigwedge \varphi. \varphi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{\} \Rightarrow \text{prv } \varphi = \text{isTrue } \varphi$   
begin  
  
theorem Tarski_semantic:  
assumes 0:  $T \in \text{fmla } \text{Fvars } T = \{xx\}$   
and 1:  $\bigwedge \varphi. \varphi \in \text{fmla} \Rightarrow \text{Fvars } \varphi = \{\} \Rightarrow \text{isTrue } (\text{eqv } (\text{subst } T \langle \varphi \rangle xx) \varphi)$   
shows  $\neg \text{consistent}$   
<proof>
```

NB: To instantiate the semantic version of Tarski's theorem for a truth predicate *isTruth* on sentences, one needs to extend it to a predicate "prv" on formulas and verify that "prv" satisfies the rules of intuitionistic logic.

end — context *Goedel_Form_prv_eq_isTrue*

Bibliography

- [1] A. Popescu and D. Traytel. A formally verified abstract account of Gödel's incompleteness theorems. In P. Fontaine, editor, *CADE 27*, volume 11716 of *LNCS*, pages 442–461. Springer, 2019.