Hall's Theorem For Countable Families of Sets and Countable Graphs

Fabián Fernando Serrano Suárez, Mauricio Ayala-Rincón, Thaynara Arielly de Lima

August 8, 2024

Contents

theory *background-on-graphs*

imports
Main

begin

1 Special Graph Theoretical Notions

This theory provides a background on specialized graph notions and properties. We follow the approach by L. Noschinski available in the AFPs. Since not all elements of Noschinski theory are required, we prefer not to import it.

The proof are desiccated in several steps since the focus is clarity instead proof automation.

record $('a,'b)$ *pre-digraph* =
  verts :: $'a$ *set*
  arcs :: $'b$ *set*
  tail :: $'b \Rightarrow 'a$
  head :: $'b \Rightarrow 'a$

definition *tails*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* **where**

tails $G \equiv \{$*tail* G e $|e.$ $e \in$ *arcs* G $\}$

definition *tails-set* :: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'b$ *set* $\Rightarrow$ $'a$ *set* **where**
tails-set G $E \equiv \{$ *tail* G e $|e.$ $e \in E \wedge E \subseteq$ *arcs* G $\}$

definition *heads*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* **where**
heads $G \equiv \{$ *head* G e $|e.$ $e \in$ *arcs* G $\}$

definition *heads-set*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'b$ *set* $\Rightarrow$ $'a$ *set* **where**
heads-set G $E \equiv \{$ *head* G e $|e.$ $e \in E \wedge E \subseteq$ *arcs* G $\}$

definition *neighbour*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a \Rightarrow$ $'a \Rightarrow$ *bool* **where**
neighbour G v u $\equiv$
$\exists e.$ $e \in$ (*arcs* G) $\wedge$ ((*head* G $e = v \wedge$ *tail* G $e = u$) $\vee$
(*head* G $e = u \wedge$ *tail* G $e = v$))

definition *neighbourhood*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ *set* **where**
neighbourhood G $v \equiv \{u$ $|u.$ *neighbour* G u $v\}$

definition *bipartite-digraph*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool* **where**
bipartite-digraph G X Y $\equiv$
$(X \cup Y =$ (*verts* G)) $\wedge$ $X \cap Y = \{\} \wedge$
($\forall e \in$ (*arcs* G).(*tail* G e) $\in X \longleftrightarrow$ (*head* G e) $\in Y$)

definition *dir-bipartite-digraph*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool*
where
dir-bipartite-digraph G X Y $\equiv$ (*bipartite-digraph* G X Y) $\wedge$
((*tails* $G = X$) $\wedge$ ($\forall$ *e1* $\in$ *arcs* G. $\forall$ *e2* $\in$ *arcs* G. *e1* $=$ *e2* $\longleftrightarrow$ *head* G *e1* $=$ *head* G *e2* $\wedge$ *tail* G *e1* $=$ *tail* G *e2*))

definition *K-E-bipartite-digraph*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool*
where
K-E-bipartite-digraph G X Y $\equiv$
(*dir-bipartite-digraph* G X Y) $\wedge$ ($\forall x \in X.$ *finite* (*neighbourhood* G x))

definition *dirBD-matching*:: $('a,'b)$ *pre-digraph* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'b$ *set* $\Rightarrow$ *bool*
where
dirBD-matching G X Y E $\equiv$
dir-bipartite-digraph G X Y $\wedge$ ($E \subseteq$ (*arcs* G)) $\wedge$
($\forall$ *e1* $\in E.$ ($\forall$ *e2* $\in E.$ *e1* $\neq$ *e2* $\longrightarrow$
((*head* G *e1*) $\neq$ (*head* G *e2*)) $\wedge$
((*tail* G *e1*) $\neq$ (*tail* G *e2*))))

lemma *tail-head*:

```
  assumes dir-bipartite-digraph G X Y and e ∈ arcs G
  shows (tail G e) ∈ X ∧ (head G e) ∈ Y
  using assms
    by (unfold dir-bipartite-digraph-def, unfold bipartite-digraph-def, unfold tails-def,
auto)

lemma tail-head1:
  assumes dirBD-matching G X Y E and e ∈ E
  shows (tail G e) ∈ X ∧ (head G e) ∈ Y
  using assms tail-head[of G X Y e] by(unfold dirBD-matching-def, auto)

lemma dirBD-matching-tail-edge-unicity:
   dirBD-matching G X Y E ⟶
    (∀ e1 ∈ E. (∀ e2∈ E. (tail G e1 = tail G e2) ⟶ e1 = e2))

proof
  assume dirBD-matching G X Y E
  thus ∀ e1∈E. ∀ e2∈E. tail G e1 = tail G e2 ⟶ e1 = e2
     by (unfold dirBD-matching-def, auto)
qed

lemma dirBD-matching-head-edge-unicity:
   dirBD-matching G X Y E ⟶
    (∀ e1 ∈ E. (∀ e2∈ E. (head G e1 = head G e2) ⟶ e1 = e2))

proof
  assume dirBD-matching G X Y E
  thus ∀ e1∈E. ∀ e2∈E. head G e1 = head G e2 ⟶ e1 = e2
     by(unfold dirBD-matching-def, auto)
qed


definition dirBD-perfect-matching::
  ('a,'b) pre-digraph ⇒ 'a set ⇒ 'a set ⇒ 'b set ⇒ bool
  where
  dirBD-perfect-matching G X Y E ≡
   dirBD-matching G X Y E ∧ (tails-set G E = X)

lemma Tail-covering-edge-in-Pef-matching:
      ∀ x∈X. dirBD-perfect-matching G X Y E ⟶ (∃ e ∈ E. tail G e = x)
proof
  fix x
  assume Hip1: x ∈ X
  show dirBD-perfect-matching G X Y E ⟶ (∃ e∈E. tail G e = x)
  proof
    assume dirBD-perfect-matching G X Y E
    hence x ∈ tails-set G E using Hip1
           by (unfold dirBD-perfect-matching-def, auto)
    thus ∃ e∈E. tail G e = x  by (unfold tails-set-def, auto)
```

qed
qed

lemma *Edge-unicity-in-dirBD-P-matching*:
$\forall x{\in}X.$ *dirBD-perfect-matching* $G\ X\ Y\ E \longrightarrow (\exists!e \in E.\ tail\ G\ e = x)$

proof
fix x
assume *Hip1*: $x \in X$
show *dirBD-perfect-matching* $G\ X\ Y\ E \longrightarrow (\exists!e \in E.\ tail\ G\ e = x)$
proof
assume *Hip2*: *dirBD-perfect-matching* $G\ X\ Y\ E$
then obtain $\exists\, e.\ e \in E \wedge tail\ G\ e = x$
using *Hip1 Tail-covering-edge-in-Pef-matching*[*of* $X\ G\ Y\ E$] **by** *auto*
then obtain e **where** e: $e \in E \wedge\ tail\ G\ e = x$ **by** *auto*
hence a: $e \in E \wedge\ tail\ G\ e = x$ **by** *auto*
show $\exists!e.\ e \in E \wedge tail\ G\ e = x$
proof
show $e \in E \wedge tail\ G\ e = x$ **using** a **by** *auto*
next
fix *e1*
assume *Hip3*: $e1 \in E \wedge tail\ G\ e1 = x$
hence $tail\ G\ e = tail\ G\ e1 \wedge e \in E\ \wedge e1 \in E$ **using** a **by** *auto*
moreover
have *dirBD-matching* $G\ X\ Y\ E$
using *Hip2* **by**(*unfold dirBD-perfect-matching-def*, *auto*)
ultimately
show $e1 = e$
using *Hip2 dirBD-matching-tail-edge-unicity*[*of* $G\ X\ Y\ E$]
by *auto*
qed
qed
qed

definition *E-head* :: $('a,'b)$ *pre-digraph* $\Rightarrow$ *'b set* $\Rightarrow ('a \Rightarrow 'a)$
where
E-head $G\ E = (\lambda x.\ (THE\ y.\ \exists\ e.\ e \in E \wedge tail\ G\ e = x \wedge\ head\ G\ e = y))$

lemma *unicity-E-head1*:
assumes *dirBD-matching* $G\ X\ Y\ E \wedge e \in E \wedge tail\ G\ e = x \wedge head\ G\ e = y$
shows $(\forall z.(\exists\ e.\ e \in E \wedge tail\ G\ e = x \wedge head\ G\ e = z) \longrightarrow z = y)$
using *assms dirBD-matching-tail-edge-unicity* **by** *blast*

lemma *unicity-E-head2*:
assumes *dirBD-matching* $G\ X\ Y\ E \wedge e \in E \wedge tail\ G\ e = x \wedge head\ G\ e = y$
shows $(THE\ a.\ \exists\ e.\ e \in E \wedge tail\ G\ e = x \wedge head\ G\ e = a) = y$
using *assms dirBD-matching-tail-edge-unicity* **by** *blast*

lemma *unicity-E-head*:

assumes *dirBD-matching G X Y E* $\wedge$ *e* $\in$ *E* $\wedge$ *tail G e* = *x* $\wedge$ *head G e* = *y*
shows (*E-head G E*) *x* = *y*
using *assms unicity-E-head2*[*of G X Y E e x y*] **by**(*unfold E-head-def*, *auto*)

lemma *E-head-image* :
dirBD-perfect-matching G X Y E $\longrightarrow$
(*e* $\in$ *E* $\wedge$ *tail G e* = *x* $\longrightarrow$ (*E-head G E*) *x* = *head G e*)

proof
assume *dirBD-perfect-matching G X Y E*
thus *e* $\in$ *E* $\wedge$ *tail G e* = *x* $\longrightarrow$ (*E-head G E*) *x* = *head G e*
using *dirBD-matching-tail-edge-unicity* [*of G X Y E*]
by (*unfold E-head-def*, *unfold dirBD-perfect-matching-def*, *blast*)
qed

lemma *E-head-in-neighbourhood*:
dirBD-matching G X Y E $\longrightarrow$ *e* $\in$ *E* $\longrightarrow$ *tail G e* = *x* $\longrightarrow$
(*E-head G E*) *x* $\in$ *neighbourhood G x*

proof (*rule impI*)+
assume
dir-BDm: *dirBD-matching G X Y E* **and** *ed*: *e* $\in$ *E* **and** *hd*: *tail G e* = *x*
show *E-head G E x* $\in$ *neighbourhood G x*
proof−
have ($\exists$ *y*. *y* = *head G e*) **using** *hd* **by** *auto*
then obtain *y* **where** *y*: *y* = *head G e* **by** *auto*
hence (*E-head G E*) *x* = *y*
using *dir-BDm ed hd unicity-E-head*[*of G X Y E e x y*]
by *auto*
moreover
have *e* $\in$ (*arcs G*) **using** *dir-BDm ed* **by**(*unfold dirBD-matching-def*, *auto*)
hence *neighbour G y x* **using** *ed hd y* **by**(*unfold neighbour-def*, *auto*)
ultimately
show *?thesis* **using** *hd ed* **by**(*unfold neighbourhood-def*, *auto*)
qed
qed

lemma *dirBD-matching-inj-on*:
dirBD-perfect-matching G X Y E $\longrightarrow$ *inj-on* (*E-head G E*) *X*

proof(*rule impI*)
assume *dirBD-pm* : *dirBD-perfect-matching G X Y E*
show *inj-on* (*E-head G E*) *X*
proof(*unfold inj-on-def*)
show $\forall$ *x*$\in$*X*. $\forall$ *y*$\in$*X*. *E-head G E x* = *E-head G E y* $\longrightarrow$ *x* = *y*
proof
fix *x*
assume *1*: *x*$\in$ *X*
show $\forall$ *y*$\in$*X*. *E-head G E x* = *E-head G E y* $\longrightarrow$ *x* = *y*

```
      proof
        fix y
        assume 2: y∈ X
        show E-head G E x = E-head G E y ⟶ x = y
        proof(rule impI)
          assume same-eheads: E-head G E x = E-head G E y
          show x=y
          proof−
            have hex: (∃!e ∈ E. tail G e = x)
              using dirBD-pm 1 Edge-unicity-in-dirBD-P-matching[of X G Y E]
              by auto
            then obtain ex where hex1: ex ∈ E ∧ tail G ex = x by auto
            have ey: (∃!e ∈ E. tail G e = y)
              using  dirBD-pm 2 Edge-unicity-in-dirBD-P-matching[of X G Y E]
              by auto
            then obtain ey where hey1: ey ∈ E ∧ tail G ey = y by auto
            have ettx: E-head G E x = head G ex
              using  dirBD-pm hex1 E-head-image[of G X Y E ex x] by auto
            have etty: E-head G E y = head G ey
              using  dirBD-pm hey1 E-head-image[of G X Y E ey y] by auto
            have same-heads: head G ex = head G ey
              using same-eheads ettx etty by auto
            hence same-edges: ex = ey
              using dirBD-pm 1 2 hex1 hey1
                    dirBD-matching-head-edge-unicity[of G X Y E]
            by(unfold dirBD-perfect-matching-def,unfold dirBD-matching-def, blast)
            thus ?thesis using  same-edges hex1 hey1 by auto
          qed
        qed
      qed
    qed
  qed
qed


end


theory Compactness

imports Main
HOL−Library.Countable-Set
ModelExistence.ModelExistence

begin
```

2 Compactness Theorem for Propositional Logic

This theory formalises the compactness theorem based on the existence model theorem. The formalisation, initially published as [2] in Spanish, was adapted to extend several combinatorial theorems over finite structures to the infinite case (e.g., see Serrano, Ayala-Rincón, and de Lima formalizations of Hall's Theorem for infinite families of sets and infinite graphs [4, 5].)

lemma *UnsatisfiableAtom*:
  shows $\neg$ (*satisfiable* {*F*, $\neg$.*F*})
proof (*rule notI*)
  assume *hip*: *satisfiable* {*F*, $\neg$.*F*}
  show *False*
  proof $-$
    have $\exists I$. *I model* {*F*, $\neg$.*F*} **using** *hip* **by**(*unfold satisfiable-def*, *auto*)
    then obtain *I* **where** *I*: (*t-v-evaluation I F*) = *Ttrue*
      and (*t-v-evaluation I* ($\neg$.*F*)) = *Ttrue*
      by(*unfold model-def*, *auto*)
    thus *False* **by**(*auto simp add*: *v-negation-def*)
  qed
qed

lemma *consistenceP-Prop1*:
  assumes $\forall$ (*A*::$'b$ *formula set*). (*A*$\subseteq$ *W* $\wedge$ *finite A*) $\longrightarrow$ *satisfiable A*
  shows ($\forall$ *P*. $\neg$ (*Atom P* $\in$ *W* $\wedge$ ($\neg$. *Atom P*) $\in$ *W*))
proof (*rule allI notI*)+
  fix *P*
  assume *h1*: *Atom P* $\in$ *W* $\wedge$ ($\neg$.*Atom P*) $\in$ *W*
  show *False*
  proof $-$
    have {*Atom P*, ($\neg$.*Atom P*)} $\subseteq$ *W* **using** *h1* **by** *simp*
    moreover
    have *finite* {*Atom P*, ($\neg$.*Atom P*)} **by** *simp*
    ultimately
    have {*Atom P*, ($\neg$.*Atom P*)} $\subseteq$ *W* $\wedge$ *finite* {*Atom P*, ($\neg$.*Atom P*)} **by** *simp*
    moreover
    have ({*Atom P*, ($\neg$.*Atom P*)}$\subseteq$ *W* $\wedge$ *finite* {*Atom P*, ($\neg$.*Atom P*)}) $\longrightarrow$
        satisfiable {*Atom P*, ($\neg$.*Atom P*)}
      using *assms* **by**(*rule-tac x* = {*Atom P*, ($\neg$.*Atom P*)} **in** *allE*, *auto*)
    ultimately
    have *satisfiable* {*Atom P*, ($\neg$.*Atom P*)} **by** *simp*
    thus *False* **using** *UnsatisfiableAtom* **by** *auto*
  qed
qed

lemma *UnsatisfiableFF*:
  shows $\neg$ (*satisfiable* {*FF*})

proof −
  have ∀ *I*. *t-v-evaluation I FF* = *Ffalse* **by** *simp*
  hence ∀ *I*. ¬ (*I model* {*FF*}) **by**(*unfold model-def*, *auto*)
  thus *?thesis* **by**(*unfold satisfiable-def*, *auto*)
qed

lemma *consistenceP-Prop2*:
  assumes ∀ (*A*::′*b formula set*). (*A*⊆ *W* ∧ *finite A*) ⟶ *satisfiable A*
  shows *FF* ∉ *W*
proof (*rule notI*)
  assume *hip*: *FF* ∈ *W*
  show *False*
  proof −
    have {*FF*} ⊆ *W* **using** *hip* **by** *simp*
    moreover
    have *finite* {*FF*} **by** *simp*
    ultimately
    have {*FF*} ⊆ *W* ∧ *finite* {*FF*} **by** *simp*
    moreover
    have ({*FF*::′*b formula*} ⊆ *W* ∧ *finite* {*FF*}) ⟶
        satisfiable {*FF*::′*b formula*}
      using *assms* **by**(*rule-tac x* = {*FF*::′*b formula*} **in** *allE*, *auto*)
    ultimately
    have *satisfiable* {*FF*::′*b formula*} **by** *simp*
    thus *False* **using** *UnsatisfiableFF* **by** *auto*
  qed
qed

lemma *UnsatisfiableFFa*:
  shows ¬ (*satisfiable* {¬.*TT*})
proof −
  have ∀ *I*. *t-v-evaluation I TT* = *Ttrue* **by** *simp*
  have ∀ *I*. *t-v-evaluation I* (¬.*TT*) = *Ffalse* **by**(*auto simp add*:*v-negation-def*)
  hence ∀ *I*. ¬ (*I model* {¬.*TT*}) **by**(*unfold model-def*, *auto*)
  thus *?thesis* **by**(*unfold satisfiable-def*, *auto*)
qed

lemma *consistenceP-Prop3*:
  assumes ∀ (*A*::′*b formula set*). (*A*⊆ *W* ∧ *finite A*) ⟶ *satisfiable A*
  shows ¬.*TT* ∉ *W*
proof (*rule notI*)
  assume *hip*: ¬.*TT* ∈ *W*
  show *False*
  proof −
    have {¬.*TT*} ⊆ *W* **using** *hip* **by** *simp*
    moreover
    have *finite* {¬.*TT*} **by** *simp*
    ultimately
    have {¬.*TT*} ⊆ *W* ∧ *finite* {¬.*TT*} **by** *simp*

moreover
have $(\{\neg.TT{::}'b\ formula\} \subseteq W \wedge finite\ \{\neg.TT\}) \longrightarrow$
$satisfiable\ \{\neg.TT{::}'b\ formula\}$
using *assms* **by**(*rule-tac* $x = \{\neg.TT{::}'b\ formula\}$ **in** *allE*, *auto*)
ultimately
have *satisfiable* $\{\neg.TT{::}'b\ formula\}$ **by** *simp*
thus *False* **using** *UnsatisfiableFFa* **by** *auto*
qed
qed

lemma *Subset-Sat*:
assumes *hip1*: *satisfiable* S **and** *hip2*: $S' \subseteq S$
shows *satisfiable* S'
proof $-$
have $\exists I.\ \forall\ F \in S.$ *t-v-evaluation* $I\ F = Ttrue$ **using** *hip1*
by (*unfold satisfiable-def*, *unfold model-def*, *auto*)
hence $\exists I.\ \forall\ F \in S'.$ *t-v-evaluation* $I\ F = Ttrue$ **using** *hip2* **by** *auto*
thus *?thesis* **by**(*unfold satisfiable-def*, *unfold model-def*, *auto*)
qed

lemma *satisfiableUnion1*:
assumes *satisfiable* $(A \cup \{\neg.\neg.F\})$
shows *satisfiable* $(A \cup \{F\})$
proof $-$
have $\exists I.\ \forall\ G \in (A \cup \{\neg.\neg.F\}).$ *t-v-evaluation* $I\ G = Ttrue$
using *assms* **by**(*unfold satisfiable-def*, *unfold model-def*, *auto*)
then obtain I **where** I: $\forall\ G \in (A \cup \{\neg.\neg.F\}).$ *t-v-evaluation* $I\ G = Ttrue$
by *auto*
hence *1*: $\forall\ G \in A.$ *t-v-evaluation* $I\ G = Ttrue$
and *2*: *t-v-evaluation* $I\ (\neg.\neg.F) = Ttrue$
by *auto*
have *typeFormula* $(\neg.\neg.F) = NoNo$ **by** *auto*
hence *t-v-evaluation* $I\ \ F = Ttrue$ **using** *EquivNoNoComp*[*of* $\neg.\neg.F$] *2*
by (*unfold equivalent-def*, *unfold Comp1-def*, *auto*)
hence $\forall\ G \in A \cup \{F\}.$ *t-v-evaluation* $I\ G = Ttrue$ **using** *1* **by** *auto*
thus *satisfiable* $(A \cup \{F\})$
by(*unfold satisfiable-def*, *unfold model-def*, *auto*)
qed

lemma *consistenceP-Prop4*:
assumes *hip1*: $\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \wedge finite\ A) \longrightarrow satisfiable\ A$
and *hip2*: $\neg.\neg.F \in W$
shows $\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \cup \{F\} \wedge finite\ A) \longrightarrow satisfiable\ A$
proof (*rule allI*, *rule impI*)+
fix A
assume *hip*: $A \subseteq W \cup \{F\} \wedge finite\ A$
show *satisfiable* A
proof $-$
have $A - \{F\} \subseteq W \wedge finite\ (A - \{F\})$ **using** *hip* **by** *auto*

hence $(A-\{F\}) \cup \{\neg.\neg.F\} \subseteq W \wedge$ *finite* $((A-\{F\}) \cup \{\neg.\neg.F\})$
using *hip2* **by** *auto*
hence *satisfiable* $((A-\{F\}) \cup \{\neg.\neg.F\})$ **using** *hip1* **by** *auto*
hence *satisfiable* $((A-\{F\}) \cup \{F\})$ **using** *satisfiableUnion1* **by** *blast*
moreover
have $A\subseteq (A-\{F\}) \cup \{F\}$ **by** *auto*
ultimately
show *satisfiable A* **using** *Subset-Sat* **by** *auto*
qed
qed

lemma *satisfiableUnion2*:
assumes *hip1*: *FormulaAlfa F* **and** *hip2*: *satisfiable* $(A \cup \{F\})$
shows *satisfiable* $(A \cup \{Comp1\ F, Comp2\ F\})$
proof $-$
have $\exists I.\forall\ G \in A \cup \{F\}.$ *t-v-evaluation I G = Ttrue*
using *hip2* **by**(*unfold satisfiable-def*, *unfold model-def*, *auto*)
then obtain *I* **where** *I*: $\forall\ G \in A \cup \{F\}.$ *t-v-evaluation I G = Ttrue* **by** *auto*

hence *1*: $\forall\ G \in A.$ *t-v-evaluation I G = Ttrue* **and** *2*: *t-v-evaluation I F = Ttrue* **by** *auto*
have *typeFormula F = Alfa* **using** *hip1 noAlfaBeta noAlfaNoNo* **by** *auto*
hence *equivalent F* $(Comp1\ F \wedge.\ Comp2\ F)$
using *2 EquivAlfaComp[of F]* **by** *auto*
hence *t-v-evaluation I* $(Comp1\ F \wedge.\ Comp2\ F) =$ *Ttrue*
using *2* **by**(*unfold equivalent-def*, *auto*)
hence *t-v-evaluation I* $(Comp1\ F) =$ *Ttrue* $\wedge$ *t-v-evaluation I* $(Comp2\ F) =$ *Ttrue*
using *ConjunctionValues* **by** *auto*
hence $\forall\ G \in A \cup \{Comp1\ F,\ Comp2\ F\}$. *t-v-evaluation I G = Ttrue* **using** *1* **by** *auto*
thus *satisfiable* $(A \cup \{Comp1\ F, Comp2\ F\})$
by (*unfold satisfiable-def*, *unfold model-def*, *auto*)
qed

lemma *consistenceP-Prop5*:
assumes *hip0*: *FormulaAlfa F*
and *hip1*: $\forall$ (*A::'b formula set*). $(A\subseteq W \wedge$ *finite A*$) \longrightarrow$ *satisfiable A*
and *hip2*: $F \in W$
shows $\forall$ (*A::'b formula set*). $(A\subseteq W \cup \{Comp1\ F,\ Comp2\ F\} \wedge$ *finite A*$) \longrightarrow$ *satisfiable A*
proof (*rule allI*, *rule impI*)+
fix *A*
assume *hip*: $A \subseteq W \cup \{Comp1\ F,\ Comp2\ F\} \wedge$ *finite A*
show *satisfiable A*
proof $-$
have $A-\{Comp1\ F,\ Comp2\ F\} \subseteq W \wedge$ *finite* $(A-\{Comp1\ F,\ Comp2\ F\})$
using *hip* **by** *auto*

```
    hence (A−{Comp1 F, Comp2 F}) ∪ {F} ⊆ W ∧
          finite ((A−{Comp1 F, Comp2 F}) ∪ {F})
      using hip2 by auto
    hence satisfiable ((A−{Comp1 F, Comp2 F}) ∪ {F})
      using hip1 by auto
    hence satisfiable ((A−{Comp1 F, Comp2 F}) ∪ {Comp1 F, Comp2 F})
      using hip0 satisfiableUnion2 by auto
    moreover
    have  A ⊆ (A−{Comp1 F, Comp2 F}) ∪ {Comp1 F, Comp2 F} by auto
    ultimately
    show satisfiable A using Subset-Sat by auto
  qed
qed


lemma satisfiableUnion3:
  assumes hip1: FormulaBeta F and hip2: satisfiable (A ∪ {F})
  shows satisfiable (A ∪ {Comp1 F}) ∨ satisfiable (A ∪ {Comp2 F})
proof −
  obtain I where I: ∀ G ∈ (A ∪ {F}). t-v-evaluation I G = Ttrue
  using hip2 by(unfold satisfiable-def, unfold model-def, auto)
  hence S1: ∀ G ∈ A. t-v-evaluation I G = Ttrue
    and S2:  t-v-evaluation I F = Ttrue
    by auto
  have V: t-v-evaluation I (Comp1 F) = Ttrue ∨ t-v-evaluation I (Comp2 F) =
Ttrue
    using hip1 S2 EquivBetaComp[of F] DisjunctionValues
    by (unfold equivalent-def, auto)
  have ((∀ G ∈ A. t-v-evaluation I G = Ttrue) ∧ t-v-evaluation I (Comp1 F) =
Ttrue) ∨
       ((∀ G ∈ A. t-v-evaluation I G = Ttrue) ∧ t-v-evaluation I (Comp2 F) =
Ttrue)
    using V
  proof (rule disjE)
    assume t-v-evaluation I (Comp1 F) = Ttrue
    hence (∀ G ∈ A. t-v-evaluation I G = Ttrue) ∧ t-v-evaluation I (Comp1 F) =
Ttrue
      using S1 by auto
    thus ?thesis by simp
  next
    assume t-v-evaluation I (Comp2 F) = Ttrue
    hence (∀ G ∈ A. t-v-evaluation I G = Ttrue) ∧ t-v-evaluation I (Comp2 F) =
Ttrue
      using S1 by auto
    thus ?thesis by simp
  qed
  hence (∀ G ∈ A ∪ {Comp1 F}. t-v-evaluation I G = Ttrue) ∨
        (∀ G ∈ A ∪ {Comp2 F}. t-v-evaluation I G = Ttrue)
    by auto
```

hence $(\exists I. \forall G \in A \cup \{Comp1\ F\}.\ t\text{-}v\text{-}evaluation\ I\ G = Ttrue) \vee$
$(\exists I. \forall G \in A \cup \{Comp2\ F\}.\ t\text{-}v\text{-}evaluation\ I\ G = Ttrue)$
by *auto*
thus *satisfiable* $(A \cup \{Comp1\ F\}) \vee$ *satisfiable* $(A \cup \{Comp2\ F\})$
by (*unfold satisfiable-def*, *unfold model-def*, *auto*)
qed

lemma *consistenceP-Prop6*:
assumes *hip0*: *FormulaBeta F*
and *hip1*: $\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \wedge finite\ A) \longrightarrow satisfiable\ A$
and *hip2*: $F \in W$
shows $(\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \cup \{Comp1\ F\} \wedge finite\ A) \longrightarrow$
$satisfiable\ A) \vee$
$(\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \cup \{Comp2\ F\} \wedge finite\ A) \longrightarrow$
$satisfiable\ A)$
proof $-$
{ assume *hip3*: $\neg((\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \cup \{Comp1\ F\} \wedge finite\ A) \longrightarrow$
$satisfiable\ A) \vee$
$(\forall\ (A{::}'b\ formula\ set).\ (A \subseteq W \cup \{Comp2\ F\} \wedge finite\ A) \longrightarrow$
$satisfiable\ A))$
have *False*
proof $-$
obtain *A B* **where** *A1*: $A \subseteq W \cup \{Comp1\ F\}$
and *A2*: *finite A*
and *A3*: $\neg$ *satisfiable A*
and *B1*: $B \subseteq W \cup \{Comp2\ F\}$
and *B2*: *finite B*
and *B3*: $\neg$ *satisfiable B*
using *hip3* **by** *auto*
have *a1*: $A - \{Comp1\ F\} \subseteq W$
and *a2*: *finite* $(A - \{Comp1\ F\})$
using *A1* **and** *A2* **by** *auto*
hence *satisfiable* $(A - \{Comp1\ F\})$ **using** *hip1* **by** *simp*
have *b1*: $B - \{Comp2\ F\} \subseteq W$
and *b2*: *finite* $(B - \{Comp2\ F\})$
using *B1* **and** *B2* **by** *auto*
hence *satisfiable* $(B - \{Comp2\ F\})$ **using** *hip1* **by** *simp*
moreover
have $(A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{F\} \subseteq W$
and *finite* $((A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{F\})$
using *a1 a2 b1 b2 hip2* **by** *auto*
hence *satisfiable* $((A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{F\})$
using *hip1* **by** *simp*
hence *satisfiable* $((A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{Comp1\ F\})$
$\vee$ *satisfiable* $((A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{Comp2\ F\})$
using *hip0 satisfiableUnion3* **by** *auto*
moreover
have $A \subseteq (A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{Comp1\ F\}$

and $B \subseteq (A - \{Comp1\ F\}) \cup (B - \{Comp2\ F\}) \cup \{Comp2\ F\}$
by *auto*
ultimately
have *satisfiable A* $\vee$ *satisfiable B* **using** *Subset-Sat* **by** *auto*
thus *False* **using** *A3 B3* **by** *simp*
qed }
thus *?thesis* **by** *auto*
qed

lemma *ConsistenceCompactness*:
shows *consistenceP*$\{W::'b\ formula\ set.\ \forall A.\ (A \subseteq W \wedge finite\ A) \longrightarrow satisfiable\ A\}$
proof (*unfold consistenceP-def*, *rule allI*, *rule impI*)
let $?C = \{W::'b\ formula\ set.\ \forall A.\ (A \subseteq W \wedge finite\ A) \longrightarrow satisfiable\ A\}$
fix $W :: 'b\ formula\ set$
assume $W \in ?C$
hence *hip*: $\forall A.\ (A \subseteq W \wedge finite\ A) \longrightarrow satisfiable\ A$ **by** *simp*
show $(\forall P.\ \neg\ (atom\ P \in W \wedge (\neg.atom\ P\ ) \in W)) \wedge$
$FF \notin W \wedge$
$\neg.TT \notin W \wedge$
$(\forall F.\ \neg.\neg.F \in W \longrightarrow W \cup \{F\} \in ?C) \wedge$
$(\forall F.\ (FormulaAlfa\ F) \wedge F \in W \longrightarrow$
$(W \cup \{Comp1\ F,\ Comp2\ F\} \in ?C)) \wedge$
$(\forall F.\ (FormulaBeta\ F) \wedge F \in W \longrightarrow$
$(W \cup \{Comp1\ F\} \in ?C \vee W \cup \{Comp2\ F\} \in ?C))$
proof −
have $(\forall P.\ \neg\ (atom\ P \in W \wedge (\neg.\ atom\ P) \in W))$
using *hip consistenceP-Prop1* **by** *simp*
moreover
have $FF \notin W$ **using** *hip consistenceP-Prop2* **by** *auto*
moreover
have $\neg.\ TT \notin W$ **using** *hip consistenceP-Prop3* **by** *auto*
moreover
have $\forall F.\ (\neg.\neg.F) \in W \longrightarrow W \cup \{F\} \in ?C$
proof (*rule allI impI*)+
fix *F*
assume *hip1*: $\neg.\neg.F \in W$
show $W \cup \{F\} \in ?C$ **using** *hip hip1 consistenceP-Prop4* **by** *simp*
qed
moreover
have
$\forall F.\ (FormulaAlfa\ F) \wedge F \in W \longrightarrow (W \cup \{Comp1\ F,\ Comp2\ F\} \in ?C)$
proof (*rule allI impI*)+
fix *F*
assume *FormulaAlfa F* $\wedge$ $F \in W$
thus $W \cup \{Comp1\ F,\ Comp2\ F\} \in ?C$ **using** *hip consistenceP-Prop5*[*of F*]
by *blast*
qed
moreover

```
    have ∀ F. (FormulaBeta F) ∧ F ∈ W ⟶
            (W ∪ {Comp1 F} ∈ ?C ∨ W ∪ {Comp2 F} ∈ ?C)
    proof (rule allI impI)+
      fix F
      assume (FormulaBeta F) ∧ F ∈ W
      thus W ∪ {Comp1 F} ∈ ?C ∨ W ∪ {Comp2 F} ∈ ?C
        using hip consistenceP-Prop6[of F] by blast
    qed
    ultimately
    show ?thesis by auto
  qed
qed

lemma countable-enumeration-formula:
  shows ∃ f. enumeration (f:: nat ⇒'a::countable formula)
  by (metis(full-types) EnumerationFormulasP1
      enumeration-def surj-def surj-from-nat)

theorem Compacteness-Theorem:
  assumes ∀ A. (A ⊆ (S:: 'a::countable formula set) ∧ finite A) ⟶ satisfiable A
  shows satisfiable S
proof −
  have enum: ∃ g. enumeration (g:: nat ⇒ 'a formula)
    using countable-enumeration-formula by auto
    let ?C = {W:: 'a formula set. ∀ A. (A ⊆ W ∧ finite A) ⟶ satisfiable A}
  have consistenceP ?C
    using ConsistenceCompactness by simp
  moreover
  have S ∈ ?C using assms by simp
  ultimately
  show satisfiable S using enum and Theo-ExistenceModels[of ?C S] by auto
qed

end

theory Hall-Theorem
  imports
   Compactness
   Marriage.Marriage
begin
```

3 Hall Theorem for countable (infinite) families of sets

Hall's Theorem for countable families of sets is proved as a consequence of compactness theorem for propositional calculus ([4]). The theory imports Marriage theory from the AFP, which proves marriage theorem for the finite case. The proof also uses an updated version of Serrano's formalization of the compactness theorem for propositional logic.

definition *system-representatives* :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow ('a \Rightarrow 'b) \Rightarrow bool$
where
system-representatives $S\ I\ R\ \equiv (\forall\, i{\in}I.\ (R\ i) \in (S\ i)) \wedge (inj\text{-}on\ R\ I)$

definition *set-to-list* :: $'a\ set \Rightarrow 'a\ list$
where *set-to-list* $s = (SOME\ l.\ set\ l = s)$

lemma *set-set-to-list*:
finite $s \Longrightarrow set\ (set\text{-}to\text{-}list\ s) = s$
unfolding *set-to-list-def* **by** (*metis* (*mono-tags*) *finite-list some-eq-ex*)

lemma *list-to-set*:
assumes *finite* $(S\ i)$
shows *set* $(set\text{-}to\text{-}list\ (S\ i)) = (S\ i)$
using *assms set-set-to-list* **by** *auto*

primrec *disjunction-atomic* :: $'b\ list \Rightarrow 'a \Rightarrow ('a \times 'b) formula$ **where**
disjunction-atomic $[]\ i = FF$
$|$ *disjunction-atomic* $(x\#D)\ i = (atom\ (i,\ x)) \vee.\ (disjunction\text{-}atomic\ D\ i)$

lemma *t-v-evaluation-disjunctions1*:
assumes *t-v-evaluation* $I\ (disjunction\text{-}atomic\ (a\ \#\ l)\ i) = Ttrue$
shows *t-v-evaluation* $I\ (atom\ (i,a)) = Ttrue \vee t\text{-}v\text{-}evaluation\ I\ (disjunction\text{-}atomic\ l\ i) = Ttrue$
proof−
have
$(disjunction\text{-}atomic\ (a\ \#\ l)\ i) = (atom\ (i,a)) \vee.\ (disjunction\text{-}atomic\ l\ i)$
by *auto*
hence *t-v-evaluation* $I\ ((atom\ (i\ ,a)) \vee.\ (disjunction\text{-}atomic\ l\ i)) = Ttrue$
using *assms* **by** *auto*
thus *?thesis* **using** *DisjunctionValues* **by** *blast*
qed

lemma *t-v-evaluation-atom*:
assumes *t-v-evaluation* $I\ (disjunction\text{-}atomic\ l\ i) = Ttrue$
shows $\exists x.\ x \in set\ l \wedge (t\text{-}v\text{-}evaluation\ I\ (atom\ (i,x)) = Ttrue)$
proof−
have *t-v-evaluation* $I\ (disjunction\text{-}atomic\ l\ i) = Ttrue \Longrightarrow$
$\exists x.\ x \in set\ l \wedge (t\text{-}v\text{-}evaluation\ I\ (atom\ (i,x)) = Ttrue)$
proof(*induct l*)
case *Nil*
then show *?case* **by** *auto*
next
case (*Cons a l*)
show $\exists x.\ x \in set\ (a\ \#\ l) \wedge t\text{-}v\text{-}evaluation\ I\ (atom\ (i,x)) = Ttrue$
proof−
have

$(t\text{-}v\text{-}evaluation\ I\ (atom\ (i,a)) = Ttrue) \vee t\text{-}v\text{-}evaluation\ I\ (disjunction\text{-}atomic\ l\ i)=Ttrue$
using *Cons(2) t-v-evaluation-disjunctions1[of I]* **by** *auto*
thus *?thesis*
proof(*rule disjE*)
assume $t\text{-}v\text{-}evaluation\ I\ (atom\ (i,a)) = Ttrue$
thus *?thesis* **by** *auto*
next
assume $t\text{-}v\text{-}evaluation\ I\ (disjunction\text{-}atomic\ l\ i) = Ttrue$
thus *?thesis* **using** *Cons* **by** *auto*
qed
qed
qed
thus *?thesis* **using** *assms* **by** *auto*
qed

definition $\mathcal{F}$ :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow (('a \times 'b)formula)\ set$ **where**
$\mathcal{F}\ S\ I \equiv (\bigcup i{\in}I.\ \{\ disjunction\text{-}atomic\ (set\text{-}to\text{-}list\ (S\ i))\ i\ \})$

definition $\mathcal{G}$ :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow ('a \times 'b)formula\ set$ **where**
$\mathcal{G}\ S\ I \equiv \{\neg.(atom\ (i,x) \wedge.\ atom(i,y))$
$|x\ y\ i\ .\ x{\in}(S\ i) \wedge y{\in}(S\ i) \wedge\ x{\neq}y \wedge i{\in}I\}$

definition $\mathcal{H}$ :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow ('a \times 'b)formula\ set$ **where**
$\mathcal{H}\ S\ I \equiv \{\neg.(atom\ (i,x) \wedge.\ atom(j,x))$
$|\ x\ i\ j.\ x \in (S\ i) \cap (S\ j) \wedge (i{\in}I \wedge j{\in}I \wedge i{\neq}j)\}$

definition $\mathcal{T}$ :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow ('a \times 'b)formula\ set$ **where**
$\mathcal{T}\ S\ I \equiv (\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I) \cup (\mathcal{H}\ S\ I)$

primrec *indices-formula* :: $('a \times 'b)formula \Rightarrow 'a\ set$ **where**
$indices\text{-}formula\ FF = \{\}$
$|\ indices\text{-}formula\ TT = \{\}$
$|\ indices\text{-}formula\ (atom\ P) = \{fst\ P\}$
$|\ indices\text{-}formula\ (\neg.\ F) = indices\text{-}formula\ F$
$|\ indices\text{-}formula\ (F \wedge.\ G) = indices\text{-}formula\ F \cup indices\text{-}formula\ G$
$|\ indices\text{-}formula\ (F \vee.\ G) = indices\text{-}formula\ F \cup indices\text{-}formula\ G$
$|\ indices\text{-}formula\ (F \rightarrow.\ G) = indices\text{-}formula\ F \cup indices\text{-}formula\ G$

definition *indices-set-formulas* :: $('a \times 'b)formula\ set \Rightarrow 'a\ set$ **where**
$indices\text{-}set\text{-}formulas\ S = (\bigcup F{\in}\ S.\ indices\text{-}formula\ F)$

lemma *finite-indices-formulas*:
shows *finite (indices-formula F)*
by(*induct F, auto*)

lemma *finite-set-indices*:
assumes *finite S*
shows *finite (indices-set-formulas S)*

using ‹*finite S*› *finite-indices-formulas*
by(*unfold indices-set-formulas-def*, *auto*)

lemma *indices-disjunction*:
assumes $F = \textit{disjunction-atomic } L \;\; i$ **and** $L \neq []$
shows *indices-formula* $F = \{i\}$
proof−
have $(F = \textit{disjunction-atomic } L \;\; i \;\wedge\; L \neq []) \Longrightarrow \textit{indices-formula } F = \{i\}$
proof(*induct L arbitrary*: *F*)
case *Nil* **hence** *False* **using** *assms* **by** *auto*
thus *?case* **by** *auto*
next
case(*Cons a L*)
assume $F = \textit{disjunction-atomic } (a \# L) \; i \wedge a \# L \neq []$
thus *?case*
proof(*cases L*)
assume $L = []$
thus *indices-formula* $F = \{i\}$ **using** *Cons*(*2*) **by** *auto*
next
show
$\bigwedge b \; \textit{list}. \; F = \textit{disjunction-atomic } (a \# L) \; i \wedge a \# L \neq [] \Longrightarrow L = b \# \textit{list} \Longrightarrow$
indices-formula $F = \{i\}$
using *Cons*(*1*−*2*) **by** *auto*
qed
qed
thus *?thesis* **using** *assms* **by** *auto*
qed

lemma *nonempty-set-list*:
assumes $\forall i{\in}I. \; (S \; i){\neq}\{\}$ **and** $\forall i{\in}I. \; \textit{finite } (S \; i)$
shows $\forall i{\in}I. \; \textit{set-to-list } (S \; i){\neq}[]$
proof(*rule ccontr*)
assume $\neg \; (\forall i{\in}I. \; \textit{set-to-list } (S \; i) \neq [])$
hence $\exists i{\in}I. \; \textit{set-to-list } (S \; i) = []$ **by** *auto*
hence $\exists i{\in}I. \; \textit{set}(\textit{set-to-list } (S \; i)) = \{\}$ **by** *auto*
then obtain *i* **where** *i*: $i{\in}I$ **and** $\textit{set } (\textit{set-to-list } (S \; i)) = \{\}$ **by** *auto*
thus *False* **using** *list-to-set*[*of S i*] *assms* **by** *auto*
qed

lemma *at-least-subset-indices*:
assumes $\forall i{\in}I. \; (S \; i){\neq}\{\}$ **and** $\forall i{\in}I. \; \textit{finite } (S \; i)$
shows *indices-set-formulas* $(\mathcal{F} \; S \; I) \subseteq I$
proof
fix *i*
assume *hip*: $i \in \textit{indices-set-formulas } (\mathcal{F} \; S \; I)$ **show** $i \in I$
proof−
have $i \in (\bigcup F{\in}(\mathcal{F} \; S \; I). \; \textit{indices-formula } F)$ **using** *hip*
by(*unfold indices-set-formulas-def*,*auto*)
hence $\exists F \in (\mathcal{F} \; S \; I). \; i \in \textit{indices-formula } F$ **by** *auto*

then obtain F **where** $F{\in}(\mathcal{F}\ S\ I)$ **and** i: $i \in$ *indices-formula* F **by** *auto*
hence $\exists\ k{\in}I.\ F =$ *disjunction-atomic* (*set-to-list* ($S\ k$)) k
by (*unfold* $\mathcal{F}$*-def*, *auto*)
then obtain k **where**
k: $k{\in}I$ **and** $F =$ *disjunction-atomic* (*set-to-list* ($S\ k$)) k **by** *auto*
hence *indices-formula* $F = \{k\}$
using *assms nonempty-set-list*[*of I S*]
indices-disjunction[*OF* ‹$F =$ *disjunction-atomic* (*set-to-list* ($S\ k$)) k›]
by *auto*
hence $k = i$ **using** i **by** *auto*
thus *?thesis* **using** k **by** *auto*
qed
qed

lemma *at-most-subset-indices*:
shows *indices-set-formulas* ($\mathcal{G}\ S\ I$) $\subseteq I$
proof
fix i
assume *hip*: $i \in$ *indices-set-formulas* ($\mathcal{G}\ S\ I$) **show** $i \in I$
proof−
have $i \in (\bigcup F{\in}(\mathcal{G}\ S\ I).$ *indices-formula* F) **using** *hip*
by(*unfold indices-set-formulas-def*,*auto*)
hence $\exists F{\in}(\mathcal{G}\ S\ I).\ i \in$ *indices-formula* F **by** *auto*
then obtain F **where** $F{\in}(\mathcal{G}\ S\ I)$ **and** i: $i \in$ *indices-formula* F
by *auto*
hence $\exists x\ y\ j.\ x{\in}(S\ j) \wedge y{\in}(S\ j) \wedge x{\neq}y \wedge j{\in}I \wedge F =$
$\neg.(atom\ (j,\ x) \wedge.\ atom(j,y))$
by (*unfold* $\mathcal{G}$*-def*, *auto*)
then obtain $x\ y\ j$ **where** $x{\in}(S\ j) \wedge y{\in}(S\ j) \wedge x{\neq}y \wedge j{\in}I$
and $F = \neg.(atom\ (j,\ x) \wedge.\ atom(j,y))$
by *auto*
hence *indices-formula* $F = \{j\} \wedge j{\in}I$ **by** *auto*
thus $i \in I$ **using** i **by** *auto*
qed
qed

lemma *different-subset-indices*:
shows *indices-set-formulas* ($\mathcal{H}\ S\ I$) $\subseteq I$
proof
fix i
assume *hip*: $i \in$ *indices-set-formulas* ($\mathcal{H}\ S\ I$) **show** $i \in I$
proof−
have $i \in (\bigcup F{\in}(\mathcal{H}\ S\ I).$ *indices-formula* F) **using** *hip*
by(*unfold indices-set-formulas-def*,*auto*)
hence $\exists F{\in}(\mathcal{H}\ S\ I)\ .\ i \in$ *indices-formula* F **by** *auto*
then obtain F **where** $F{\in}(\mathcal{H}\ S\ I)$ **and** i: $i \in$ *indices-formula* F
by *auto*
hence $\exists\ x\ j\ k.\ x \in (S\ j) \cap (S\ k) \wedge (j{\in}I \wedge k{\in}I \wedge j{\neq}k) \wedge F =$
$\neg.(atom\ (j,x) \wedge.\ atom(k,x))$

```
      by (unfold ℋ-def, auto)
    then obtain x j k
      where (j∈I ∧ k∈I ∧ j≠k) ∧ F = ¬.(atom (j, x) ∧. atom(k,x))
      by auto
    hence u: j∈I and v: k∈I and indices-formula F = {j,k}
      by auto
    hence i=j ∨ i=k  using i by auto
    thus i ∈ I using u v  by auto
  qed
qed

lemma indices-union-sets:
 shows indices-set-formulas(A ∪ B) = (indices-set-formulas A) ∪ (indices-set-formulas
B)
   by(unfold  indices-set-formulas-def, auto)

lemma at-least-subset-subset-indices1:
  assumes F∈(ℱ S I)
  shows (indices-formula F) ⊆ (indices-set-formulas (ℱ S I))
proof
  fix i
  assume hip: i ∈ indices-formula F
  show  i ∈ indices-set-formulas (ℱ S I)
  proof−
    have ∃F. F∈(ℱ S I) ∧ i ∈ indices-formula F using assms hip by auto
    thus  ?thesis by(unfold indices-set-formulas-def, auto)
  qed
qed

lemma at-most-subset-subset-indices1:
  assumes  F∈(𝒢 S I)
  shows (indices-formula F) ⊆ (indices-set-formulas (𝒢 S I))
proof
  fix i
  assume hip: i ∈ indices-formula F
  show  i ∈ indices-set-formulas (𝒢 S I)
  proof−
    have ∃F. F∈(𝒢 S I) ∧ i ∈ indices-formula F using assms hip by auto
    thus ?thesis by(unfold indices-set-formulas-def, auto)
  qed
qed

lemma different-subset-indices1:
  assumes  F∈(ℋ S I)
  shows (indices-formula F) ⊆ (indices-set-formulas (ℋ S I))
proof
  fix i
  assume hip: i ∈ indices-formula F
  show  i ∈ indices-set-formulas (ℋ S I)
```

proof−
have $\exists F.\ F \in (\mathcal{H}\ S\ I) \wedge i \in$ *indices-formula* F **using** *assms hip* **by** *auto*
thus *?thesis* **by**(*unfold indices-set-formulas-def*, *auto*)
qed
qed

lemma *all-subset-indices*:
assumes $\forall i \in I.(S\ i) \neq \{\}$ **and** $\forall i \in I.$ *finite*$(S\ i)$
shows *indices-set-formulas* $(\mathcal{T}\ S\ I) \subseteq I$
proof
fix i
assume *hip*: $i \in$ *indices-set-formulas* $(\mathcal{T}\ S\ I)$ **show** $i \in I$
proof−
have $i \in$ *indices-set-formulas* $((\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I) \cup (\mathcal{H}\ S\ I))$
using *hip* **by** (*unfold* $\mathcal{T}$*-def*,*auto*)
hence $i \in$ *indices-set-formulas* $((\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I)) \cup$
indices-set-formulas$(\mathcal{H}\ S\ I)$
using *indices-union-sets*[*of* $(\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I)$] **by** *auto*
hence $i \in$ *indices-set-formulas* $((\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I)) \vee$
$i \in$ *indices-set-formulas*$(\mathcal{H}\ S\ I)$
by *auto*
thus *?thesis*
proof(*rule disjE*)
assume *hip*: $i \in$ *indices-set-formulas* $(\mathcal{F}\ S\ I \cup \mathcal{G}\ S\ I)$
hence $i \in (\bigcup F \in (\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I).$ *indices-formula* $F)$
by(*unfold indices-set-formulas-def*, *auto*)
then obtain F
where F: $F \in (\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I)$ **and** i: $i \in$ *indices-formula* F **by** *auto*
from F **have** (*indices-formula* F) $\subseteq$ (*indices-set-formulas* $(\mathcal{F}\ S\ I)$)
$\vee$ *indices-formula* $F \subseteq$ (*indices-set-formulas* $(\mathcal{G}\ S\ I)$)
using *at-least-subset-subset-indices1 at-most-subset-subset-indices1* **by** *blast*
hence $i \in$ *indices-set-formulas* $(\mathcal{F}\ S\ I) \vee$
$i \in$ *indices-set-formulas* $(\mathcal{G}\ S\ I)$
using i **by** *auto*
thus $i \in I$
using *assms at-least-subset-indices*[*of I S*] *at-most-subset-indices*[*of S I*] **by**
auto
next
assume $i \in$ *indices-set-formulas* $(\mathcal{H}\ S\ I)$
hence
$i \in (\bigcup F \in (\mathcal{H}\ S\ I).$ *indices-formula* $F)$
by(*unfold indices-set-formulas-def*, *auto*)
then obtain F **where** F: $F \in (\mathcal{H}\ S\ I)$ **and** i: $i \in$ *indices-formula* F
by *auto*
from F **have** (*indices-formula* F) $\subseteq$ (*indices-set-formulas* $(\mathcal{H}\ S\ I)$)
using *different-subset-indices1* **by** *blast*
hence $i \in$ *indices-set-formulas* $(\mathcal{H}\ S\ I)$ **using** i **by** *auto*
thus $i \in I$ **using** *different-subset-indices*[*of S I*]
by *auto*

    qed
  qed
qed

lemma *inclusion-indices*:
  assumes $S \subseteq H$
  shows *indices-set-formulas* $S \subseteq$ *indices-set-formulas* H
proof
  fix i
  assume $i \in$ *indices-set-formulas* S
  hence $\exists F.\ F \in S \wedge i \in$ *indices-formula* F
    by(*unfold indices-set-formulas-def*, *auto*)
  hence $\exists F.\ F \in H \wedge i \in$ *indices-formula* F **using** *assms* **by** *auto*
  thus $i \in$ *indices-set-formulas* H
    by(*unfold indices-set-formulas-def*, *auto*)
qed

lemma *indices-subset-formulas*:
  assumes $\forall i{\in}I.(S\ i){\neq}\{\}$ **and** $\forall i{\in}I.$ *finite*$(S\ i)$ **and** $A \subseteq (\mathcal{T}\ S\ I)$
  shows (*indices-set-formulas* A) $\subseteq I$
proof−
  have (*indices-set-formulas* A) $\subseteq$ (*indices-set-formulas* $(\mathcal{T}\ S\ I)$)
    using *assms*(*3*) *inclusion-indices* **by** *auto*
  thus *?thesis* **using** *assms*(*1*−*2*) *all-subset-indices*[*of I S*] **by** *auto*
qed

lemma *To-subset-all-its-indices*:
  assumes $\forall i{\in}I.\ (S\ i){\neq}\{\}$ **and** $\forall i{\in}I.$ *finite* $(S\ i)$ **and** $To \subseteq (\mathcal{T}\ S\ I)$
  shows $To \subseteq (\mathcal{T}\ S$ (*indices-set-formulas* To))
proof
  fix F
  assume *hip*: $F \in To$
  hence $F \in (\mathcal{T}\ S\ I)$ **using** *assms*(*3*) **by** *auto*
  hence $F \in (\mathcal{F}\ S\ I) \cup (\mathcal{G}\ S\ I) \cup (\mathcal{H}\ S\ I)$ **by**(*unfold* $\mathcal{T}$*-def*,*auto*)
  hence $F \in (\mathcal{F}\ S\ I) \vee F \in (\mathcal{G}\ S\ I) \vee F \in (\mathcal{H}\ S\ I)$ **by** *auto*
  thus $F{\in}(\mathcal{T}\ S$ (*indices-set-formulas* To))
  proof(*rule disjE*)
    assume $F \in (\mathcal{F}\ S\ I)$
    hence $\exists i{\in}I.\ F =$ *disjunction-atomic* (*set-to-list* $(S\ i)$) i
      by(*unfold* $\mathcal{F}$*-def*,*auto*)
    then obtain i
      where i: $i{\in}I$ **and** F: $F =$ *disjunction-atomic* (*set-to-list* $(S\ i)$) i
      by *auto*
    hence *indices-formula* $F = \{i\}$
      using
       assms(*1*−*2*) *nonempty-set-list*[*of I S*] *indices-disjunction*[*of F* (*set-to-list* (*S*
i)) *i*]
      by *auto*
    hence $i{\in}$(*indices-set-formulas* To) **using** *hip*

by(*unfold indices-set-formulas-def*,*auto*)
hence $F\in(\mathcal{F}\ S\ (\textit{indices-set-formulas To}))$
using F **by**(*unfold* $\mathcal{F}$*-def*,*auto*)
thus $F\in(\mathcal{T}\ S\ (\textit{indices-set-formulas To}))$
by(*unfold* $\mathcal{T}$*-def*,*auto*)
next
assume $F \in (\mathcal{G}\ S\ I) \vee F \in (\mathcal{H}\ S\ I)$
thus *?thesis*
proof(*rule disjE*)
assume $F \in (\mathcal{G}\ S\ I)$
hence $\exists\, x.\exists\, y.\exists\, i.\ F = \neg.(\textit{atom}\ (i,x) \wedge.\ \textit{atom}(i,y)) \wedge x\in(S\ i) \wedge$
$y\in(S\ i) \wedge\ \ x\neq y \wedge i\in I$
by(*unfold* $\mathcal{G}$*-def*, *auto*)
then obtain $x\ y\ i$
where $F1$: $F = \neg.(\textit{atom}\ (i,x) \wedge.\ \textit{atom}(i,y))$ **and**
$F2$: $x\in(S\ i) \wedge y\in(S\ i) \wedge\ \ x\neq y \wedge i\in I$
by *auto*
hence *indices-formula* $F = \{i\}$ **by** *auto*
hence $i\in(\textit{indices-set-formulas To})$ **using** *hip*
by(*unfold indices-set-formulas-def*,*auto*)
hence $F\in(\mathcal{G}\ S\ (\textit{indices-set-formulas To}))$
using *F1 F2* **by**(*unfold* $\mathcal{G}$*-def*,*auto*)
thus $F\in(\mathcal{T}\ S\ (\textit{indices-set-formulas To}))$ **by**(*unfold* $\mathcal{T}$*-def*,*auto*)
next
assume $F \in (\mathcal{H}\ S\ I)$
hence $\exists\, x.\exists\, i.\exists\, j.\ F = \neg.(\textit{atom}\ (i,x) \wedge.\ \textit{atom}(j,x)) \wedge$
$x \in (S\ i) \cap (S\ j) \wedge (i\in I \wedge j\in I \wedge i\neq j)$
by(*unfold* $\mathcal{H}$*-def*, *auto*)
then obtain $x\ i\ j$
where $F3$: $F = \neg.(\textit{atom}\ (i,x) \wedge.\ \textit{atom}(j,x))$ **and**
$F4$: $x \in (S\ i) \cap (S\ j) \wedge (i\in I \wedge j\in I \wedge i\neq j)$
by *auto*
hence *indices-formula* $F = \{i,j\}$ **by** *auto*
hence $i\in(\textit{indices-set-formulas To}) \wedge j\in(\textit{indices-set-formulas To})$
using *hip* **by**(*unfold indices-set-formulas-def*,*auto*)
hence $F\in(\mathcal{H}\ S\ (\textit{indices-set-formulas To}))$
using *F3 F4* **by**(*unfold* $\mathcal{H}$*-def*,*auto*)
thus $F\in(\mathcal{T}\ S\ (\textit{indices-set-formulas To}))$ **by**(*unfold* $\mathcal{T}$*-def*,*auto*)
qed
qed
qed

lemma *all-nonempty-sets*:
assumes $\forall\, i\in I.\ (S\ i)\neq\{\}$ **and** $\forall\, i\in I.\ \textit{finite}\ (S\ i)$ **and** $A \subseteq (\mathcal{T}\ S\ I)$
shows $\forall\, i\in(\textit{indices-set-formulas}\ A).\ (S\ i)\neq\{\}$
proof−
have $(\textit{indices-set-formulas}\ A)\subseteq I$
using *assms*(*1*−*3*) *indices-subset-formulas*[*of I S A*] **by** *auto*
thus *?thesis* **using** *assms*(*1*) **by** *auto*

qed

lemma *all-finite-sets*:
  assumes $\forall i{\in}I.\ (S\ i){\neq}\{\}$ **and** $\forall i{\in}I.$ *finite* $(S\ i)$ **and** $A \subseteq (\mathcal{T}\ S\ I)$
shows $\forall i{\in}($*indices-set-formulas* $A).$ *finite* $(S\ i)$
proof−
  have (*indices-set-formulas* $A){\subseteq}I$
    using *assms*(*1*−*3*) *indices-subset-formulas*[*of I S A*] **by** *auto*
  thus $\forall i{\in}($*indices-set-formulas* $A).$ *finite* $(S\ i)$ **using** *assms*(*2*) **by** *auto*
qed

lemma *all-nonempty-sets1*:
  assumes $\forall J{\subseteq}I.$ *finite* $J \longrightarrow$ *card* $J \leq$ *card* $(\bigcup\ (S\ `\ J))$
  shows $\forall i{\in}I.\ (S\ i){\neq}\{\}$ **using** *assms* **by** *auto*

lemma *system-distinct-representatives-finite*:
  assumes
  $\forall i{\in}I.\ (S\ i){\neq}\{\}$ **and** $\forall i{\in}I.$ *finite* $(S\ i)$ **and** *To* $\subseteq (\mathcal{T}\ S\ I)$ **and** *finite To*
   and $\forall J{\subseteq}($*indices-set-formulas To*$).$ *card* $J \leq$ *card* $(\bigcup\ (S\ `\ J))$
 shows $\exists R.$ *system-representatives S* (*indices-set-formulas To*) *R*
proof−
  have *1*: *finite* (*indices-set-formulas To*)
    using *assms*(*4*) *finite-set-indices* **by** *auto*
  have $\forall i{\in}($*indices-set-formulas To*$).$ *finite* $(S\ i)$
    using *all-finite-sets assms*(*1*−*3*) **by** *auto*
  hence $\exists R.\ (\forall i{\in}($*indices-set-formulas To*$).\ R\ i \in S\ i) \wedge$
          inj-on R (*indices-set-formulas To*)
    using *1 assms*(*5*) *marriage-HV*[*of* (*indices-set-formulas To*) *S*] **by** *auto*
  then obtain *R*
    where *R*: $(\forall i{\in}($*indices-set-formulas To*$).\ R\ i \in S\ i) \wedge$
          inj-on R (*indices-set-formulas To*) **by** *auto*
  thus *?thesis* **by**(*unfold system-representatives-def*, *auto*)
qed

fun *Hall-interpretation* :: $('a \Rightarrow 'b\ set) \Rightarrow 'a\ set \Rightarrow ('a \Rightarrow 'b) \Rightarrow (('a \times 'b) \Rightarrow$ *v-truth*) **where**
Hall-interpretation $A\ \mathcal{I}\ R = (\lambda(i,x).($*if* $i \in \mathcal{I} \wedge x \in (A\ i) \wedge (R\ i) = x$ *then Ttrue else Ffalse*))

lemma *t-v-evaluation-index*:
  assumes *t-v-evaluation* (*Hall-interpretation S I R*) (*atom* $(i,x)) =$ *Ttrue*
  shows $(R\ i) = x$
proof(*rule ccontr*)
  assume $(R\ i) \neq x$ **hence** *t-v-evaluation* (*Hall-interpretation S I R*) (*atom* $(i,x))$ $\neq$ *Ttrue*
    by *auto*
  hence *t-v-evaluation* (*Hall-interpretation S I R*) (*atom* $(i,x)) =$ *Ffalse*
  using *non-Ttrue*[*of Hall-interpretation S I R atom* (i,x)] **by** *auto*
  thus *False* **using** *assms* **by** *simp*

qed

lemma *distinct-elements-distinct-indices*:
assumes $F = \neg.(atom\ (i,x) \wedge.\ atom(i,y))$ **and** $x \neq y$
shows *t-v-evaluation* (*Hall-interpretation S I R*) F = *Ttrue*
proof(*rule ccontr*)
assume *t-v-evaluation* (*Hall-interpretation S I R*) $F \neq$ *Ttrue*
hence
t-v-evaluation (*Hall-interpretation S I R*) ($\neg.(atom\ (i,x) \wedge.\ atom\ (i,\ y))$) $\neq$ *Ttrue*

using *assms*(*1*) **by** *auto*
hence
t-v-evaluation (*Hall-interpretation S I R*) ($\neg.(atom\ (i,x) \wedge.\ atom\ (i,\ y))$) = *Ffalse*
using
non-Ttrue[*of Hall-interpretation S I R* $\neg.(atom\ (i,x) \wedge.\ atom\ (i,\ y))$]
by *auto*
hence *t-v-evaluation* (*Hall-interpretation S I R*) ($(atom\ (i,x) \wedge.\ atom\ (i,\ y))$) = *Ttrue*
using
NegationValues1[*of Hall-interpretation S I R* $(atom\ (i,x) \wedge.\ atom\ (i,\ y))$]
by *auto*
hence *t-v-evaluation* (*Hall-interpretation S I R*) ($atom\ (i,x)$) = *Ttrue* **and**
t-v-evaluation (*Hall-interpretation S I R*) ($atom\ (i,\ y)$) = *Ttrue*
using
ConjunctionValues[*of Hall-interpretation S I R* $atom\ (i,x)$ $atom\ (i,\ y)$]
by *auto*
hence $(R\ i) = x$ **and** $(R\ i) = y$ **using** *t-v-evaluation-index* **by** *auto*
hence $x=y$ **by** *auto*
thus *False* **using** *assms*(*2*) **by** *auto*
qed

lemma *same-element-same-index*:
assumes
$F = \neg.(atom\ (i,x) \wedge.\ atom(j,x))$ **and** $i \in I \wedge j \in I$ **and** $i \neq j$ **and** *inj-on R I*
shows *t-v-evaluation* (*Hall-interpretation S I R*) F = *Ttrue*
proof(*rule ccontr*)
assume *t-v-evaluation* (*Hall-interpretation S I R*) $F \neq$ *Ttrue*
hence *t-v-evaluation* (*Hall-interpretation S I R*) ($\neg.(atom\ (i,x) \wedge.\ atom\ (j,x))$) $\neq$ *Ttrue*
using *assms*(*1*) **by** *auto*
hence
t-v-evaluation (*Hall-interpretation S I R*) ($\neg.(atom\ (i,x) \wedge.\ atom\ (j,\ x))$) = *Ffalse*
using
non-Ttrue[*of Hall-interpretation S I R* $\neg.(atom\ (i,x) \wedge.\ atom\ (j,\ x))$]
by *auto*
hence *t-v-evaluation* (*Hall-interpretation S I R*) ($(atom\ (i,x) \wedge.\ atom\ (j,\ x))$) = *Ttrue*
using
NegationValues1[*of Hall-interpretation S I R* $(atom\ (i,x) \wedge.\ atom\ (j,\ x))$]

    by *auto*
  hence *t-v-evaluation* (*Hall-interpretation S I R*) (*atom* (*i*,*x*)) = *Ttrue* **and**
  t-v-evaluation (*Hall-interpretation S I R*) (*atom* (*j*, *x*)) = *Ttrue*
    using *ConjunctionValues*[*of Hall-interpretation S I R atom* (*i*,*x*) *atom* (*j*,*x*)]
    by *auto*
  hence (*R i*)= *x* **and** (*R j*)= *x* **using** *t-v-evaluation-index* **by** *auto*
  hence (*R i*) = (*R j*) **by** *auto*
  hence *i*=*j* **using** ‹*i*∈*I* ∧ *j*∈*I*› ‹*inj-on R I*› **by**(*unfold inj-on-def*, *auto*)
  thus *False* **using** ‹*i*≠*j*› **by** *auto*
qed

lemma *disjunctor-Ttrue-in-atomic-disjunctions*:
  assumes *x* ∈ *set l* **and** *t-v-evaluation I* (*atom* (*i*,*x*)) = *Ttrue*
  shows *t-v-evaluation I* (*disjunction-atomic l i*) = *Ttrue*
proof−
  have *x* ∈ *set l* ⟹ *t-v-evaluation I* (*atom* (*i*,*x*)) = *Ttrue* ⟹
  t-v-evaluation I (*disjunction-atomic l i*) = *Ttrue*
  proof(*induct l*)
    case *Nil*
    then show *?case* **by** *auto*
  next
    case (*Cons a l*)
    then show *t-v-evaluation I* (*disjunction-atomic* (*a* # *l*) *i*) = *Ttrue*
    proof−
      have *x* = *a* ∨ *x*≠*a* **by** *auto*
      thus *t-v-evaluation I* (*disjunction-atomic* (*a* # *l*) *i*) = *Ttrue*
      proof(*rule disjE*)
        assume *x* = *a*
          hence
          1:(*disjunction-atomic* (*a*#*l*) *i*) =
            (*atom* (*i*,*x*)) ∨. (*disjunction-atomic l i*)
          by *auto*
        have *t-v-evaluation I* ((*atom* (*i*,*x*)) ∨. (*disjunction-atomic l i*)) = *Ttrue*
        using *Cons*(*3*) **by**(*unfold t-v-evaluation-def*,*unfold v-disjunction-def*, *auto*)
        thus *?thesis* **using** *1* **by** *auto*
      next
        assume *x* ≠ *a*
        hence *x*∈ *set l* **using** *Cons*(*2*) **by** *auto*
        hence *t-v-evaluation I* (*disjunction-atomic l i*) = *Ttrue*
          using *Cons*(*1*) *Cons*(*3*) **by** *auto*
        thus *?thesis*
          by(*unfold t-v-evaluation-def*,*unfold v-disjunction-def*, *auto*)
      qed
    qed
  qed
  thus *?thesis* **using** *assms* **by** *auto*
qed

lemma *t-v-evaluation-disjunctions*:

assumes *finite* $(S\ i)$
and $x \in (S\ i) \ \wedge\ $ *t-v-evaluation* I (*atom* (i,x)) $=$ *Ttrue*
and $F =$ *disjunction-atomic* (*set-to-list* $(S\ i)$) i
shows *t-v-evaluation* $I\ F =$ *Ttrue*
proof−
have *set* (*set-to-list* $(S\ i)$) $= (S\ i)$
using *set-set-to-list assms*(*1*) **by** *auto*
hence $x \in$ *set* (*set-to-list* $(S\ i)$)
using *assms*(*2*) **by** *auto*
thus *t-v-evaluation* $I\ F =$ *Ttrue*
using *assms*(*2*−*3*) *disjunctor-Ttrue-in-atomic-disjunctions* **by** *auto*
qed

theorem *SDR-satisfiable*:
assumes $\forall i \in \mathcal{I}.\ (A\ i) \neq \{\}$ **and** $\forall i \in \mathcal{I}.$ *finite* $(A\ i)$ **and** $X \subseteq (\mathcal{T}\ A\ \mathcal{I})$
and *system-representatives* $A\ \mathcal{I}\ R$
shows *satisfiable* X
proof−
have *satisfiable* $(\mathcal{T}\ A\ \mathcal{I})$
proof−
have *inj-on* $R\ \mathcal{I}$ **using** *assms*(*4*) *system-representatives-def*[*of* $A\ \mathcal{I}\ R$] **by** *auto*
have (*Hall-interpretation* $A\ \mathcal{I}\ R$) *model* $(\mathcal{T}\ A\ \mathcal{I})$
proof(*unfold model-def*)
show $\forall F \in (\mathcal{T}\ A\ \mathcal{I}).$ *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) $F =$ *Ttrue*
proof
fix F **assume** $F \in (\mathcal{T}\ A\ \mathcal{I})$
show *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) $F =$ *Ttrue*
proof−
have $F \in (\mathcal{F}\ A\ \mathcal{I}) \cup (\mathcal{G}\ A\ \mathcal{I}) \cup (\mathcal{H}\ A\ \mathcal{I})$
using ‹$F \in (\mathcal{T}\ A\ \mathcal{I})$› *assms*(*3*) **by**(*unfold* $\mathcal{T}$*-def*,*auto*)
hence $F \in (\mathcal{F}\ A\ \mathcal{I}) \vee F \in (\mathcal{G}\ A\ \mathcal{I}) \vee F \in (\mathcal{H}\ A\ \mathcal{I})$ **by** *auto*
thus *?thesis*
proof(*rule disjE*)
assume $F \in (\mathcal{F}\ A\ \mathcal{I})$
hence $\exists i \in \mathcal{I}.\ F =$ *disjunction-atomic* (*set-to-list* $(A\ i)$) i
by(*unfold* $\mathcal{F}$*-def*,*auto*)
then obtain i
where i: $i \in \mathcal{I}$ **and** F: $F =$ *disjunction-atomic* (*set-to-list* $(A\ i)$) i
by *auto*
have *1*: *finite* $(A\ i)$ **using** i *assms*(*2*) **by** *auto*
have *2*: $i \in \mathcal{I} \wedge (R\ i) \in (A\ i)$
using i *assms*(*4*) **by** (*unfold system-representatives-def*, *auto*)
hence *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) (*atom* $(i,(R\ i))$) $=$
Ttrue
by *auto*
thus *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) $F =$ *Ttrue*
using *1 2 assms*(*4*) F
t-v-evaluation-disjunctions
[*of* $A\ i\ (R\ i)$ (*Hall-interpretation* $A\ \mathcal{I}\ R$) F]

by *auto*
next
assume $F \in (\mathcal{G}\ A\ \mathcal{I}) \vee F \in (\mathcal{H}\ A\ \mathcal{I})$
thus *?thesis*
proof(*rule disjE*)
assume $F \in (\mathcal{G}\ A\ \mathcal{I})$
hence
$\exists x.\exists y.\exists i.\ F = \neg.(atom\ (i,x) \wedge.\ atom(i,y)) \wedge x{\in}(A\ i) \wedge$
$y{\in}(A\ i) \wedge\ \ x{\neq}y \wedge i{\in}\mathcal{I}$
by(*unfold* $\mathcal{G}$-*def*, *auto*)
then obtain $x\ y\ i$
where F: $F = \neg.(atom\ (i,x) \wedge.\ atom(i,y))$
and $x{\in}(A\ i) \wedge y{\in}(A\ i) \wedge\ \ x{\neq}y \wedge i{\in}\mathcal{I}$
by *auto*
thus *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) $F\ = Ttrue$
using ‹*inj-on* $R\ \mathcal{I}$› *distinct-elements-distinct-indices*[*of* $F\ i\ x\ y\ A\ \mathcal{I}\ R$]
by *auto*
next
assume $F \in (\mathcal{H}\ A\ \mathcal{I})$
hence $\exists x.\exists i.\exists j.\ F = \neg.(atom\ (i,x) \wedge.\ atom(j,x)) \wedge$
$x \in (A\ i) \cap (A\ j) \wedge (i{\in}\mathcal{I} \wedge j{\in}\mathcal{I} \wedge i{\neq}j)$
by(*unfold* $\mathcal{H}$-*def*, *auto*)
then obtain $x\ i\ j$
where $F = \neg.(atom\ (i,x) \wedge.\ atom(j,x))$ **and** $(i{\in}\mathcal{I} \wedge j{\in}\mathcal{I} \wedge i{\neq}j)$
by *auto*
thus *t-v-evaluation* (*Hall-interpretation* $A\ \mathcal{I}\ R$) $F\ = Ttrue$ **using**
‹*inj-on* $R\ \mathcal{I}$›
same-element-same-index[*of* $F\ i\ x\ j\ \mathcal{I}$] **by** *auto*
qed
qed
qed
qed
qed
thus *satisfiable* $(\mathcal{T}\ A\ \mathcal{I})$ **by**(*unfold satisfiable-def*, *auto*)
qed
thus *satisfiable X* **using** *satisfiable-subset assms*(*3*) **by** *auto*
qed

lemma *finite-is-satisfiable*:
assumes
$\forall i{\in}I.\ (S\ i){\neq}\{\}$ **and** $\forall i{\in}I.$ *finite* $(S\ i)$ **and** $To \subseteq (\mathcal{T}\ S\ I)$ **and** *finite To*
and $\forall J{\subseteq}$(*indices-set-formulas To*). *card* $J \leq$ *card* $(\bigcup\ (S\ `\ J))$
shows *satisfiable To*
proof−
have *0*: $\exists R.$ *system-representatives S* (*indices-set-formulas To*) R
using *assms system-distinct-representatives-finite*[*of I S To*] **by** *auto*
then obtain R
where R: *system-representatives S* (*indices-set-formulas To*) R **by** *auto*
have *1*: $\forall i{\in}$(*indices-set-formulas To*). $(S\ i){\neq}\{\}$

using *assms*(*1*−*3*) *all-nonempty-sets* **by** *blast*
have *2*: ∀ *i*∈(*indices-set-formulas To*). *finite* (*S i*)
using *assms*(*1*−*3*) *all-finite-sets* **by** *blast*
have *3*: *To*⊆(*𝒯 S* (*indices-set-formulas To*))
using *assms*(*1*−*3*) *To-subset-all-its-indices*[*of I S To*] **by** *auto*
thus *satisfiable To*
using *1 2 3 0 SDR-satisfiable* **by** *auto*
qed

lemma *diag-nat*:
shows ∀ *y z*.∃ *x*. (*y*,*z*) = *diag x*
using *enumeration-natxnat* **by**(*unfold enumeration-def*,*auto*)

lemma *EnumFormulasHall*:
assumes ∃ *g*. *enumeration* (*g*:: *nat* ⇒′*a*) **and** ∃ *h*. *enumeration* (*h*:: *nat* ⇒′*b*)
shows ∃ *f*. *enumeration* (*f*:: *nat* ⇒(′*a* ×′*b*)*formula*)
proof−
from *assms*(*1*) **obtain** *g* **where** *e1*: *enumeration* (*g*:: *nat* ⇒′*a*) **by** *auto*
from *assms*(*2*) **obtain** *h* **where** *e2*: *enumeration* (*h*:: *nat* ⇒′*b*) **by** *auto*
have *enumeration* ((λ*m*.(*g*(*fst*(*diag m*)),(*h*(*snd*(*diag m*))))):: *nat* ⇒(′*a* ×′*b*))
proof(*unfold enumeration-def*)
show ∀ *y*::(′*a* × ′*b*). ∃ *m*. *y* = (*g* (*fst* (*diag m*)), *h* (*snd* (*diag m*)))
proof
fix *y*::(′*a* ×′*b*)
show ∃ *m*. *y* = (*g* (*fst* (*diag m*)), *h* (*snd* (*diag m*)))
proof−
have *y* = ((*fst y*), (*snd y*)) **by** *auto*
from *e1* **have** ∀ *w*::′*a*. ∃ *n1*. *w* = (*g n1*) **by**(*unfold enumeration-def*, *auto*)
hence ∃ *n1*. (*fst y*) = (*g n1*) **by** *auto*
then obtain *n1* **where** *n1*: (*fst y*) = (*g n1*) **by** *auto*
from *e2* **have** ∀ *w*::′*b*. ∃ *n2*. *w* = (*h n2*) **by**(*unfold enumeration-def*, *auto*)
hence ∃ *n2*. (*snd y*) = (*h n2*) **by** *auto*
then obtain *n2* **where** *n2*: (*snd y*) = (*h n2*) **by** *auto*
have ∃ *m*. (*n1*, *n2*) = *diag m* **using** *diag-nat* **by** *auto*
hence ∃ *m*. (*n1*, *n2*) = (*fst* (*diag m*), *snd* (*diag m*)) **by** *simp*
hence ∃ *m*.((*fst y*), (*snd y*)) = (*g*(*fst* (*diag m*)), *h*(*snd* (*diag m*)))
using *n1 n2* **by** *blast*
thus ∃ *m*. *y* = (*g* (*fst* (*diag m*)), *h*(*snd* (*diag m*))) **by** *auto*
qed
qed
qed
thus ∃ *f*. *enumeration* (*f*:: *nat* ⇒(′*a* ×′*b*)*formula*)
using *EnumerationFormulasP1* **by** *auto*
qed

theorem *all-formulas-satisfiable*:
fixes *S* :: (′*a*::*countable* ⇒ ′*b*::*countable set*) **and** *I* :: ′*a set*
assumes ∀ *i*∈(*I*::′*a set*). *finite* (*S i*) **and** ∀ *J*⊆*I*. *finite J* ⟶ *card J* ≤ *card* (⋃ (*S* ‘ *J*))

shows *satisfiable* ($\mathcal{T}$ *S I*)
proof−
  have ∀ *A*. *A* ⊆ ($\mathcal{T}$ *S I*) ∧ (*finite A*) ⟶ *satisfiable A*
  proof(*rule allI*, *rule impI*)
    fix *A* **assume** *A* ⊆ ($\mathcal{T}$ *S I*) ∧ (*finite A*)
    hence *hip1*: *A* ⊆ ($\mathcal{T}$ *S I*) **and** *hip2*: *finite A* **by** *auto*
    show *satisfiable A*
    proof −
      have *0*: ∀ *i*∈*I*. (*S i*)≠{} **using** *assms*(*2*) *all-nonempty-sets1* **by** *auto*
      hence *1*: (*indices-set-formulas A*)⊆*I*
        using *assms*(*1*) *hip1 indices-subset-formulas*[*of I S A*] **by** *auto*
      have *2*: *finite* (*indices-set-formulas A*)
        using *hip2 finite-set-indices* **by** *auto*
     have *3*: *card* (*indices-set-formulas A*) ≤ *card*(⋃ (*S* ‘(*indices-set-formulas A*)))
        using *1 2 assms*(*2*) **by** *auto*
      have ∀ *J*⊆(*indices-set-formulas A*). *card J* ≤ *card*(⋃ (*S* ‘ *J*))
     proof(*rule allI*)
       fix *J*
       show *J* ⊆ *indices-set-formulas A* ⟶ *card J* ≤ *card* (⋃ (*S* ‘ *J*))
       proof(*rule impI*)
         assume *hip*: *J*⊆(*indices-set-formulas A*)
       hence *4*: *finite J*
         using *2 rev-finite-subset* **by** *auto*
       have *J*⊆*I* **using** *hip 1* **by** *auto*
       thus *card J* ≤ *card* (⋃ (*S* ‘ *J*)) **using** *4 assms*(*2*) **by** *auto*
     qed
   qed
   thus *satisfiable A*
     using *0 assms*(*1*) *hip1 hip2 finite-is-satisfiable*[*of I S A*] **by** *auto*
 qed
qed
  thus *satisfiable* ($\mathcal{T}$ *S I*) **using**
  Compacteness-Theorem[*of* ($\mathcal{T}$ *S I*)]
    by *auto*
qed

fun *SDR* :: ((′*a* × ′*b*) ⇒ *v-truth*) ⇒ (′*a* ⇒ ′*b set*) ⇒ ′*a set* ⇒ (′*a* ⇒′*b*)
  where
SDR M S I = (λ*i*. (*THE x*. (*t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue*) ∧ *x*∈(*S i*)))

lemma *existence-representants*:
 assumes *i* ∈ *I* **and** *M model* ($\mathcal{F}$ *S I*) **and** *finite*(*S i*)
  shows ∃ *x*. (*t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue*) ∧ *x* ∈ (*S i*)
proof−
  from ‹*i* ∈ *I*›
  have (*disjunction-atomic* (*set-to-list* (*S i*)) *i*) ∈ ($\mathcal{F}$ *S I*)
    by(*unfold* $\mathcal{F}$-*def*,*auto*)
  hence *t-v-evaluation M* (*disjunction-atomic*(*set-to-list* (*S i*)) *i*) = *Ttrue*
    using *assms*(*2*) *model-def*[*of M* $\mathcal{F}$ *S I*] **by** *auto*

hence *1*: $\exists x.\ x \in set\ (set\text{-}to\text{-}list\ (S\ i)) \land (t\text{-}v\text{-}evaluation\ M\ (atom\ (i,x)) = Ttrue)$
using *t-v-evaluation-atom*[*of M* (*set-to-list* (*S i*)) *i*] **by** *auto*
thus $\exists x.\ (t\text{-}v\text{-}evaluation\ M\ (atom\ (i,x)) = Ttrue) \land\ x \in (S\ i)$
using ‹*finite*(*S i*)› *set-set-to-list*[*of* (*S i*)] **by** *auto*
qed

lemma *unicity-representants*:
shows $\forall y.(x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I) \longrightarrow$
$(\neg.(atom\ (i,x) \land.\ atom(i,y)){\in}\ (\mathcal{G}\ S\ I))$
proof(*rule allI*)
fix *y*
show $x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I \longrightarrow$
$(\neg.(atom\ (i,x) \land.\ atom(i,y)){\in}\ (\mathcal{G}\ S\ I))$
proof(*rule impI*)
assume $x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I$
thus $\neg.(atom\ (i,x) \land.\ atom(i,y)) \in (\mathcal{G}\ S\ I)$
by(*unfold* $\mathcal{G}$*-def*, *auto*)
qed
qed

lemma *unicity-selection-representants*:
assumes $i \in I$ **and** *M model* ($\mathcal{G}$ *S I*)
shows $\forall y.(x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I) \longrightarrow$
$(t\text{-}v\text{-}evaluation\ M\ (\neg.(atom\ (i,x) \land.\ atom(i,y))) = Ttrue)$
proof−
have $\forall y.(x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I) \longrightarrow$
$(\neg.(atom\ (i,x) \land.\ atom(i,y)){\in}\ (\mathcal{G}\ S\ I))$
using *unicity-representants*[*of x S i*] **by** *auto*
thus $\forall y.(x{\in}(S\ i) \land y{\in}(S\ i) \land\ x{\neq}y \land i{\in}I) \longrightarrow$
$(t\text{-}v\text{-}evaluation\ M\ (\neg.(atom\ (i,x) \land.\ atom(i,y))) = Ttrue)$
using *assms*(*2*) *model-def*[*of M* $\mathcal{G}$ *S I*] **by** *blast*
qed

lemma *uniqueness-satisfaction*:
assumes $t\text{-}v\text{-}evaluation\ M\ (atom\ (i,x)) = Ttrue \land x{\in}(S\ i)$ **and**
$\forall y.\ y \in (S\ i) \land x{\neq}y\ \longrightarrow\ t\text{-}v\text{-}evaluation\ M\ (atom\ (i,\ y)) = Ffalse$
shows $\forall z.\ t\text{-}v\text{-}evaluation\ M\ (atom\ (i,\ z)) = Ttrue \land z{\in}(S\ i) \longrightarrow z = x$
proof(*rule allI*)
fix *z*
show $t\text{-}v\text{-}evaluation\ M\ (atom\ (i,\ z)) = Ttrue \land z \in S\ i\ \longrightarrow z = x$
proof(*rule impI*)
assume *hip*: $t\text{-}v\text{-}evaluation\ M\ (atom\ (i,\ z)) = Ttrue \land z \in (S\ i)$
show $z = x$
proof(*rule ccontr*)
assume *1*: $z \neq x$
have *2*: $z \in (S\ i)$ **using** *hip* **by** *auto*
hence $t\text{-}v\text{-}evaluation\ M\ (atom(i,z)) =\ Ffalse$ **using** *1 assms*(*2*) **by** *auto*
thus *False* **using** *hip* **by** *auto*
qed

qed
qed

lemma *uniqueness-satisfaction-in-Si*:
assumes *t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue* ∧ *x*∈(*S i*) **and**
∀ *y*. *y* ∈ (*S i*) ∧ *x*≠*y* ⟶ (*t-v-evaluation M* (¬.(*atom* (*i*,*x*) ∧. *atom*(*i*,*y*))) = *Ttrue*)
shows ∀ *y*. *y* ∈ (*S i*) ∧ *x*≠*y* ⟶ *t-v-evaluation M* (*atom* (*i*, *y*)) = *Ffalse*
proof(*rule allI*, *rule impI*)
fix *y*
assume *hip*: *y* ∈ *S i* ∧ *x* ≠ *y*
show *t-v-evaluation M* (*atom* (*i*, *y*)) = *Ffalse*
proof(*rule ccontr*)
assume *t-v-evaluation M* (*atom* (*i*, *y*)) ≠ *Ffalse*
hence *t-v-evaluation M* (*atom* (*i*, *y*)) = *Ttrue* **using** *Bivaluation* **by** *blast*
hence *1*: *t-v-evaluation M* (*atom* (*i*,*x*) ∧. *atom*(*i*,*y*)) = *Ttrue*
using *assms*(*1*) *v-conjunction-def* **by** *auto*
have *t-v-evaluation M* (¬.(*atom* (*i*,*x*) ∧. *atom*(*i*,*y*))) = *Ttrue*
using *hip assms*(*2*) **by** *auto*
hence *t-v-evaluation M* (*atom* (*i*,*x*) ∧. *atom*(*i*,*y*)) = *Ffalse*
using *NegationValues2* **by** *blast*
thus *False* **using** *1* **by** *auto*
qed
qed

lemma *uniqueness-aux1*:
assumes *t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue* ∧ *x*∈(*S i*)
and ∀ *y*. *y* ∈ (*S i*) ∧ *x*≠*y* ⟶ (*t-v-evaluation M* (¬.(*atom* (*i*,*x*) ∧. *atom*(*i*,*y*))) = *Ttrue*)
shows ∀ *z*. *t-v-evaluation M* (*atom* (*i*, *z*)) = *Ttrue* ∧ *z*∈(*S i*) ⟶ *z* = *x*
using *assms uniqueness-satisfaction-in-Si*[*of M i x*] *uniqueness-satisfaction*[*of M i x*] **by** *blast*

lemma *uniqueness-aux2*:
assumes *t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue* ∧ *x*∈(*S i*) **and**
(⋀*z*.(*t-v-evaluation M* (*atom* (*i*, *z*)) = *Ttrue* ∧ *z*∈(*S i*)) ⟹ *z* = *x*)
shows (*THE a*. (*t-v-evaluation M* (*atom* (*i*,*a*)) = *Ttrue*) ∧ *a*∈(*S i*)) = *x*
using *assms* **by**(*rule the-equality*)

lemma *uniqueness-aux*:
assumes *t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue* ∧ *x*∈(*S i*) **and**
∀ *y*. *y* ∈ (*S i*) ∧ *x*≠*y* ⟶ (*t-v-evaluation M* (¬.(*atom* (*i*,*x*) ∧. *atom*(*i*,*y*))) = *Ttrue*)
shows (*THE a*. (*t-v-evaluation M* (*atom* (*i*,*a*)) = *Ttrue*) ∧ *a*∈(*S i*)) = *x*
using *assms uniqueness-aux1*[*of M i x*] *uniqueness-aux2*[*of M i x*] **by** *blast*

lemma *function-SDR*:
assumes *i* ∈ *I* **and** *M model* ($\mathcal{F}$ *S I*) **and** *M model* ($\mathcal{G}$ *S I*) **and** *finite*(*S i*)
shows ∃!*x*. (*t-v-evaluation M* (*atom* (*i*,*x*)) = *Ttrue*) ∧ *x* ∈ (*S i*) ∧ (*SDR M S I*

$i) = x$
proof−
  have $\exists x.$ (*t-v-evaluation M* (*atom* (i,x)) = *Ttrue*) $\wedge$ $x \in (S\ i)$
    using *assms*(*1*−*2*,*4*) *existence-representants* **by** *auto*
  then obtain x **where** x: (*t-v-evaluation M* (*atom* (i,x)) = *Ttrue*) $\wedge$ $x \in (S\ i)$
    by *auto*
  moreover
  have $\forall y.(x \in (S\ i) \wedge y \in (S\ i) \wedge x \neq y \wedge i \in I) \longrightarrow$
  (*t-v-evaluation M* ($\neg$.(*atom* (i,x) $\wedge$. *atom*(i,y)))) = *Ttrue*)
    using *assms*(*1*,*3*) *unicity-selection-representants*[*of i I M S*] **by** *auto*
  hence (*THE a.* (*t-v-evaluation M* (*atom* (i,a)) = *Ttrue*) $\wedge$ $a \in (S\ i)$) = x
   using x ‹$i \in I$› *uniqueness-aux*[*of M i x*] **by** *auto*
  hence *SDR M S I i* = x **by** *auto*
  hence (*t-v-evaluation M* (*atom* (i,x)) = *Ttrue* $\wedge$ $x \in (S\ i)$) $\wedge$ *SDR M S I i* = x
    using x **by** *auto*
  thus *?thesis* **by** *auto*
qed

lemma *aux-for-$\mathcal{H}$-formulas*:
  assumes
  (*t-v-evaluation M* (*atom* (i,a)) = *Ttrue*) $\wedge$ $a \in (S\ i)$
  and (*t-v-evaluation M* (*atom* (j,b)) = *Ttrue*) $\wedge$ $b \in (S\ j)$
  and $i \in I \wedge j \in I \wedge i \neq j$
  and $(a \in (S\ i) \cap (S\ j) \wedge i \in I \wedge j \in I \wedge i \neq j \longrightarrow$
  (*t-v-evaluation M* ($\neg$.(*atom* (i,a) $\wedge$. *atom*(j,a)))) = *Ttrue*))
  shows $a \neq b$
proof(*rule ccontr*)
  assume $\neg\ a \neq b$
  hence *hip*: $a=b$ **by** *auto*
  hence *t-v-evaluation M* (*atom* $(i,\ a)$) = *Ttrue* **and** *t-v-evaluation M* (*atom* $(j,$
$a)$) = *Ttrue*
    using *assms* **by** *auto*
 hence *t-v-evaluation M* (*atom* $(i,\ a)$ $\wedge$. *atom*(j,a)) = *Ttrue* **using** *v-conjunction-def*
    by *auto*
  hence *t-v-evaluation M* ($\neg$.(*atom* $(i,\ a)$ $\wedge$. *atom*(j,a)))) = *Ffalse*
    using *v-negation-def* **by** *auto*
  moreover
  have $a \in (S\ i) \cap (S\ j)$ **using** *hip assms*(*1*−*2*) **by** *auto*
  hence *t-v-evaluation M* ($\neg$.(*atom* $(i,\ a)$ $\wedge$. *atom*$(j,\ a)$))) = *Ttrue*
    using *assms*(*3*−*4*) **by** *auto*
  ultimately show *False* **by** *auto*
qed

lemma *model-of-all*:
  assumes *M model* ($\mathcal{T}$ *S I*)
  shows *M model* ($\mathcal{F}$ *S I*) **and** *M model* ($\mathcal{G}$ *S I*) **and** *M model* ($\mathcal{H}$ *S I*)
proof(*unfold model-def*)
  show $\forall F \in \mathcal{F}$ *S I*. *t-v-evaluation M F* = *Ttrue*
  proof

fix F
assume $F\in (\mathcal{F}\ S\ I)$ **hence** $F\in(\mathcal{T}\ S\ I)$ **by**(*unfold* $\mathcal{T}$*-def*, *auto*)
thus *t-v-evaluation* $M\ F$ = *Ttrue* **using** *assms* **by**(*unfold model-def*, *auto*)
qed
next
show $\forall F\in(\mathcal{G}\ S\ I)$. *t-v-evaluation* $M\ F$ = *Ttrue*
proof
fix F
assume $F\in(\mathcal{G}\ S\ I)$ **hence** $F\in(\mathcal{T}\ S\ I)$ **by**(*unfold* $\mathcal{T}$*-def*, *auto*)
thus *t-v-evaluation* $M\ F$ = *Ttrue* **using** *assms* **by**(*unfold model-def*, *auto*)
qed
next
show $\forall F\in(\mathcal{H}\ S\ I)$. *t-v-evaluation* $M\ F$ = *Ttrue*
proof
fix F
assume $F\in(\mathcal{H}\ S\ I)$ **hence** $F\in(\mathcal{T}\ S\ I)$ **by**(*unfold* $\mathcal{T}$*-def*, *auto*)
thus *t-v-evaluation* $M\ F$ = *Ttrue* **using** *assms* **by**(*unfold model-def*, *auto*)
qed
qed

lemma *sets-have-distinct-representants*:
assumes
hip1: $i\in I$ **and** *hip2*: $j\in I$ **and** *hip3*: $i\neq j$ **and** *hip4*: *M model* $(\mathcal{T}\ S\ I)$
and *hip5*: *finite*$(S\ i)$ **and** *hip6*: *finite*$(S\ j)$
shows *SDR M S I i* $\neq$ *SDR M S I j*
proof−
have *1*: *M model* $\mathcal{F}$ *S I* **and** *2*: *M model* $\mathcal{G}$ *S I*
using *hip4 model-of-all* **by** *auto*
hence $\exists!x$. (*t-v-evaluation M* (*atom* (i,x)) = *Ttrue*) $\wedge$ $x \in (S\ i)$ $\wedge$ *SDR M S I* $i = x$
using *hip1 hip4 hip5 function-SDR*[*of i I M S*] **by** *auto*
then obtain x **where**
x1: (*t-v-evaluation M* (*atom* (i,x)) = *Ttrue*) $\wedge$ $x \in (S\ i)$ **and** *x2*: *SDR M S I i* $= x$
by *auto*
have $\exists!y$. (*t-v-evaluation M* (*atom* (j,y)) = *Ttrue*) $\wedge$ $y \in (S\ j)$ $\wedge$ *SDR M S I j* $= y$
using *1 2 hip2 hip4 hip6 function-SDR*[*of j I M S*] **by** *auto*
then obtain y **where**
y1: (*t-v-evaluation M* (*atom* (j,y)) = *Ttrue*) $\wedge$ $y \in (S\ j)$ **and** *y2*: *SDR M S I j* $= y$
by *auto*
have $(x \in (S\ i) \cap (S\ j) \wedge i\in I \wedge j\in I \wedge i\neq j) \longrightarrow$
$(\neg.($*atom* (i,x) $\wedge.$ *atom*$(j,x))\in (\mathcal{H}\ S\ I))$
by(*unfold* $\mathcal{H}$*-def*, *auto*)
hence $(x \in (S\ i) \cap (S\ j) \wedge i\in I \wedge j\in I \wedge i\neq j) \longrightarrow$
$(\neg.($*atom* (i,x) $\wedge.$ *atom*$(j,x)) \in (\mathcal{T}\ S\ I))$
by(*unfold* $\mathcal{T}$*-def*, *auto*)
hence $(x \in (S\ i) \cap (S\ j) \wedge i\in I \wedge j\in I \wedge i\neq j) \longrightarrow$

$(t\text{-}v\text{-}evaluation\ M\ (\neg.(atom\ (i,x)\ \wedge.\ atom(j,x))) = Ttrue)$
  using *hip4 model-def*[*of* $M\ \mathcal{T}\ S\ I$] **by** *auto*
 hence $x \neq y$ **using** *x1 y1 assms*(*1−3*) *aux-for-$\mathcal{H}$-formulas*[*of* $M\ i\ x\ \ S\ \ j\ y\ I$]
  by *auto*
 thus *?thesis* **using** *x2 y2* **by** *auto*
qed

lemma *satisfiable-representant*:
 assumes *satisfiable* $(\mathcal{T}\ S\ I)$ **and** $\forall\, i{\in}I.$ *finite* $(S\ i)$
 shows $\exists\, R.$ *system-representatives* $S\ I\ R$
proof−
 from *assms* **have** $\exists\, M.\ M\ model\ (\mathcal{T}\ S\ I)$ **by**(*unfold satisfiable-def*)
 then obtain M **where** M: $M\ model\ (\mathcal{T}\ S\ I)$ **by** *auto*
 hence *system-representatives* $S\ I\ (SDR\ M\ S\ I)$
 proof(*unfold system-representatives-def*)
  show $(\forall\, i{\in}I.\ (SDR\ M\ S\ I\ i) \in (S\ i)) \wedge$ *inj-on* $(SDR\ M\ S\ I)\ I$
  proof(*rule conjI*)
   show $\forall\, i{\in}I.\ (SDR\ M\ S\ I\ i) \in (S\ i)$
   proof
    fix i
    assume i: $i \in I$
    have $M\ model\ \mathcal{F}\ S\ I$ **and** *2*: $M\ model\ \mathcal{G}\ S\ I$ **using** *M model-of-all*
     by *auto*
    thus $(SDR\ M\ S\ I\ i) \in (S\ i)$
     using *i M assms*(*2*) *model-of-all*[*of* $M\ S\ I$]
         function-SDR[*of* $i\ I\ M\ S$] **by** *auto*
   qed
  next
   show *inj-on* $(SDR\ M\ S\ I)\ I$
   proof(*unfold inj-on-def*)
    show $\forall\, i{\in}I.\ \forall\, j{\in}I.\ SDR\ M\ S\ I\ i = SDR\ M\ S\ I\ j \longrightarrow i = j$
    proof
     fix i
     assume *1*: $i \in I$
     show $\forall\, j{\in}I.\ SDR\ M\ S\ I\ i = SDR\ M\ S\ I\ j \longrightarrow i = j$
     proof
      fix j
      assume *2*: $j \in I$
      show $SDR\ M\ S\ I\ i = SDR\ M\ S\ I\ j \longrightarrow i = j$
      proof(*rule ccontr*)
       assume $\neg\ (SDR\ M\ S\ I\ i = SDR\ M\ S\ I\ j \longrightarrow i = j)$
       hence *5*: $SDR\ M\ S\ I\ i = SDR\ M\ S\ I\ j$ **and** *6*: $i \neq j$ **by** *auto*
       have *3*: *finite*$(S\ i)$ **and** *4*: *finite*$(S\ j)$ **using** *1 2 assms*(*2*) **by** *auto*
       have $SDR\ M\ S\ I\ i \neq SDR\ M\ S\ I\ j$
        using *1 2 3 4 6 M sets-have-distinct-representants*[*of* $i\ I\ j\ M\ S$] **by**
auto
       thus *False* **using** *5* **by** *auto*
      qed
     qed

```
        qed
      qed
    qed
  qed
  thus ∃R. system-representatives S I R by auto
qed

theorem Hall:
  fixes S :: ('a::countable ⇒ 'b::countable set) and I :: 'a set
  assumes Finite: ∀i∈I. finite (S i)
  and Marriage: ∀J⊆I. finite J ⟶ card J ≤ card (⋃ (S ` J))
 shows ∃R. system-representatives S I R
proof−
  have satisfiable (𝒯 S I) using assms all-formulas-satisfiable[of I] by auto
  thus ?thesis using Finite Marriage satisfiable-representant[of S I] by auto
qed

theorem marriage-necessity:
  fixes A :: 'a ⇒ 'b set and I :: 'a set
  assumes ∀ i∈I. finite (A i)
  and ∃R. (∀i∈I. R i ∈ A i) ∧ inj-on R I (is ∃R. ?R R A & ?inj R A)
  shows ∀J⊆I. finite J ⟶ card J ≤ card (⋃(A ` J))
proof clarify
  fix J
  assume J ⊆ I and 1: finite J
  show card J ≤ card (⋃(A ` J))
  proof−
    from assms(2) obtain R where ?R R A and ?inj R A by auto
    have inj-on R J by(rule subset-inj-on[OF ‹?inj R A› ‹J⊆I›])
    moreover have (R ` J) ⊆ (⋃(A ` J)) using ‹J⊆I› ‹?R R A› by auto
    moreover have finite (⋃(A ` J)) using ‹J⊆I› 1 assms
      by auto
    ultimately show ?thesis by (rule card-inj-on-le)
  qed
qed

end

theory Hall-Theorem-Graphs
  imports
          background-on-graphs
          HOL−Library.Countable-Set
          Hall-Theorem


begin
```

4 Hall Theorem for countable (infinite) Graphs

This section formalizes Hall Theorem for countable infinite Graphs ([5]). The proof applied a proof of Hall's theorem for countable infinite families of sets, obtained by the authors directly from the compactness theorem for propositional logic. The proof is based on Smullyan's approach given in the third chapter of his influential textbook on mathematical logic [3], based on Henkin's model existence theorem. It follows the impeccable presentation in Fitting's textbook [1].

```
definition dirBD-to-Hall::
  ('a,'b) pre-digraph ⇒ 'a set ⇒ 'a set ⇒ 'a set ⇒ ('a  ⇒ 'a set) ⇒ bool
  where
  dirBD-to-Hall G X Y I S ≡
  dir-bipartite-digraph G X Y ∧ I = X ∧ (∀ v∈I. (S v) = (neighbourhood G v))


theorem dir-BD-to-Hall:
  dirBD-perfect-matching G X Y E ⟶
   system-representatives (neighbourhood G) X (E-head G E)
proof(rule impI)
 assume dirBD-pm :dirBD-perfect-matching G X Y E
 show system-representatives (neighbourhood G) X (E-head G E)
 proof−
   have wS : dirBD-to-Hall G X Y X (neighbourhood G)
   using dirBD-pm
   by(unfold dirBD-to-Hall-def,unfold dirBD-perfect-matching-def,
      unfold dirBD-matching-def, auto)
   have arc: E ⊆ arcs G using dirBD-pm
     by(unfold dirBD-perfect-matching-def, unfold dirBD-matching-def,auto)
   have a: ∀ i. i ∈ X ⟶ E-head G E i ∈ neighbourhood G i
   proof(rule allI)
     fix i
     show i ∈ X ⟶ E-head G E i ∈ neighbourhood G i
     proof
       assume 1: i ∈ X
       show E-head G E i ∈ neighbourhood G i
       proof−
         have 2: ∃!e ∈ E. tail G e = i
         using 1 dirBD-pm Edge-unicity-in-dirBD-P-matching [of X G Y E ]
          by auto
         then obtain e where 3: e ∈ E ∧ tail G e = i by auto
       thus E-head G E i ∈ neighbourhood G i
         using  dirBD-pm 1 3 E-head-in-neighbourhood[of G X Y E e i]
         by (unfold dirBD-perfect-matching-def, auto)
       qed
     qed
   qed
   thus system-representatives (neighbourhood G) X (E-head G E)
   using a dirBD-pm dirBD-matching-inj-on [of G X Y E]
```

by (*unfold system-representatives-def*, *auto*)
qed
qed

lemma *marriage-necessary-graph*:
assumes (*dirBD-perfect-matching G X Y E*) **and** $\forall i{\in}X$. *finite* (*neighbourhood G i*)
shows $\forall J{\subseteq}X$. *finite J* $\longrightarrow$ (*card J*) $\leq$ *card* ($\bigcup$ (*neighbourhood G ‘ J*))
proof(*rule allI*, *rule impI*)
fix *J*
assume *hip1*: $J \subseteq X$
show *finite J* $\longrightarrow$ *card J* $\leq$ *card* ($\bigcup$ (*neighbourhood G ‘ J*))
proof
assume *hip2*: *finite J*
show *card J* $\leq$ *card* ($\bigcup$ (*neighbourhood G ‘ J*))
proof−
have $\exists R$. ($\forall i{\in}X$. *R i* $\in$ *neighbourhood G i*) $\wedge$ *inj-on R X*
using *assms dir-BD-to-Hall*[*of G X Y E*]
by(*unfold system-representatives-def*, *auto*)
thus *?thesis* **using** *assms*(*2*) *marriage-necessity*[*of X neighbourhood G*] *hip1 hip2* **by** *auto*
qed
qed
qed

lemma *neighbour3*:
fixes *G* :: ($'a$, $'b$) *pre-digraph* **and** *X*:: $'a$ *set*
assumes *dir-bipartite-digraph G X Y* **and** $x{\in}X$
shows *neighbourhood G x* = $\{y \mid y.\ \exists e.\ e \in$ *arcs G* $\wedge$ ((x = *tail G e*) $\wedge$ (y = *head G e*))$\}$
proof
show *neighbourhood G x* $\subseteq \{y \mid y.\ \exists e.\ e \in$ *arcs G* $\wedge$ x = *tail G e* $\wedge$ y = *head G e*$\}$
proof
fix *z*
assume *hip*: $z \in$ *neighbourhood G x*
show $z \in \{y \mid y.\ \exists e.\ e \in$ *arcs G* $\wedge$ x = *tail G e* $\wedge$ y = *head G e*$\}$
proof−
have *neighbour G z x* **using** *hip* **by**(*unfold neighbourhood-def*, *auto*)
hence $\exists e.\ e \in$ *arcs G* $\wedge$((z = (*head G e*) $\wedge$ x =(*tail G e*) $\vee$
((x = (*head G e*) $\wedge$ z = (*tail G e*)))))
using *assms* **by** (*unfold neighbour-def*, *auto*)
hence $\exists e.\ e \in$ *arcs G* $\wedge$ (z = (*head G e*) $\wedge$ x =(*tail G e*))
using *assms*
by(*unfold dir-bipartite-digraph-def*, *unfold bipartite-digraph-def*, *unfold tails-def*, *blast*)
thus *?thesis* **by** *auto*

```
    qed
  qed
  next
  show {y |y. ∃ e. e ∈ arcs G ∧ x = tail G e ∧ y = head G e} ⊆ neighbourhood G
x
  proof
    fix z
    assume hip1: z ∈ {y |y. ∃ e. e ∈ arcs G ∧ x = tail G e ∧ y = head G e}
    thus  z ∈ neighbourhood G x
      by(unfold neighbourhood-def, unfold neighbour-def, auto)
  qed
qed

lemma perfect:
  fixes  G :: ('a, 'b) pre-digraph and X:: 'a set
  assumes dir-bipartite-digraph G X Y and system-representatives (neighbourhood
G) X R
  shows  tails-set G {e |e. e ∈ (arcs G) ∧ ((tail G e) ∈ X ∧ (head G e) = R(tail
G e))} = X
proof(unfold tails-set-def)
  let ?E = {e |e. e ∈ (arcs G) ∧ ((tail G e) ∈ X ∧ (head G e) = R (tail G e))}
  show  {tail G e |e. e ∈ ?E ∧ ?E ⊆ arcs G} = X
  proof
    show {tail G e |e. e ∈ ?E ∧ ?E ⊆ arcs G}⊆ X
    proof
      fix x
      assume hip1: x ∈ {tail G e |e. e ∈ ?E ∧ ?E ⊆ arcs G}
      show x∈X
      proof−
        have ∃ e. x = tail G e ∧ e ∈ ?E ∧ ?E ⊆ arcs G using hip1 by auto
        then obtain e where e: x = tail G e ∧ e ∈ ?E ∧ ?E ⊆ arcs G by auto
        thus x∈X
          using assms(1) by(unfold dir-bipartite-digraph-def, unfold tails-def, auto)
      qed
    qed
    next
    show X ⊆ {tail G e |e. e ∈ ?E ∧ ?E ⊆ arcs G}
    proof
      fix x
      assume hip2: x∈X
      show x∈{tail G e |e. e ∈ ?E ∧ ?E ⊆ arcs G}
      proof−
        have R (x) ∈ neighbourhood G x
          using assms(2) hip2 by (unfold system-representatives-def, auto)
        hence ∃ e. e∈ arcs G ∧ (x = tail G e ∧ R(x) = (head G e))
          using assms(1) hip2 neighbour3[of G  X Y] by auto
        moreover
        have  ?E ⊆ arcs G by auto
        ultimately show ?thesis
```

```
        using hip2 assms(1) by(unfold dir-bipartite-digraph-def, unfold tails-def,
auto)
      qed
    qed
  qed
qed

lemma dirBD-matching:
  fixes  G :: ('a, 'b) pre-digraph and X:: 'a set
 assumes dir-bipartite-digraph G X Y and R: system-representatives (neighbourhood
G) X R
  and  e1 ∈ arcs G ∧ tail G e1 ∈ X and e2 ∈ arcs G ∧ tail G e2 ∈ X
  and  R(tail G e1) = head G e1
  and  R(tail G e2) = head G e2
shows e1 ≠ e2 ⟶ head G e1 ≠ head G e2 ∧ tail G e1 ≠ tail G e2
proof
  assume hip: e1 ≠ e2
  show head G e1 ≠ head G e2 ∧ tail G e1 ≠ tail G e2
  proof−
    have  (e1 = e2) = (head G e1 = head G e2 ∧ tail G e1 = tail G e2)
      using assms(1)  assms(3−4)  by(unfold dir-bipartite-digraph-def, auto)
    hence 1:  tail G e1 = tail G e2 ⟶ head G e1 ≠ head G e2
      using hip assms(1)  by auto
    have  2:  tail G e1 = tail G e2 ⟶ head G e1 = head  G e2
      using  assms(1−2) assms(5−6)  by auto
    have 3: tail G e1 ≠ tail G e2
    proof(rule notI)
      assume ∗:  tail G e1 = tail G e2
      thus False using 1 2 by auto
    qed
    have 4:  tail G e1 ≠ tail G e2 ⟶ head G e1 ≠ head G e2
      proof
        assume ∗∗: tail G e1 ≠ tail G e2
        show  head G e1 ≠ head G e2
          using ∗∗ assms(3−6) R  inj-on-def[of R X]
          system-representatives-def[of (neighbourhood G) X R] by auto
      qed
    thus ?thesis using 3 by auto
  qed
qed

lemma marriage-sufficiency-graph:
  fixes  G :: ('a::countable, 'b::countable) pre-digraph and X:: 'a set
  assumes dir-bipartite-digraph G X Y  and ∀ i∈X. finite (neighbourhood G i)
  shows
  (∀ J⊆X. finite J ⟶ (card J) ≤ card (⋃ (neighbourhood G ‘ J))) ⟶
   (∃ E. dirBD-perfect-matching G X Y E)
proof(rule impI)
  assume hip: ∀ J⊆X. finite J ⟶ card J ≤ card (⋃ (neighbourhood G ‘ J))
```

show $\exists E.$ *dirBD-perfect-matching G X Y E*
proof−
have $\exists R.$ *system-representatives* (*neighbourhood G*) *X R*
using *assms hip Hall*[*of X neighbourhood G*] **by** *auto*
then obtain *R* **where** *R*: *system-representatives* (*neighbourhood G*) *X R* **by** *auto*
let *?E* = $\{e \mid e.\ e \in (arcs\ G) \wedge ((tail\ G\ e) \in X \wedge (head\ G\ e) = R\ (tail\ G\ e))\}$
have *dirBD-perfect-matching G X Y ?E*
proof(*unfold dirBD-perfect-matching-def*, *rule conjI*)
show *dirBD-matching G X Y ?E*
proof(*unfold dirBD-matching-def*, *rule conjI*)
show *dir-bipartite-digraph G X Y* **using** *assms*(*1*) **by** *auto*
next
show *?E* $\subseteq$ *arcs G* $\wedge$ ($\forall$ *e1*$\in$*?E*. $\forall$ *e2*$\in$*?E*.
e1 $\neq$ *e2* $\longrightarrow$ *head G e1* $\neq$ *head G e2* $\wedge$ *tail G e1* $\neq$ *tail G e2*)
proof(*rule conjI*)
show *?E* $\subseteq$ *arcs G* **by** *auto*
next
show $\forall$ *e1*$\in$*?E*. $\forall$ *e2*$\in$*?E*. *e1* $\neq$ *e2* $\longrightarrow$ *head G e1* $\neq$ *head G e2* $\wedge$ *tail G e1* $\neq$ *tail G e2*
proof
fix *e1*
assume *H1*: *e1* $\in$ *?E*
show $\forall$ *e2*$\in$ *?E*. *e1* $\neq$ *e2* $\longrightarrow$ *head G e1* $\neq$ *head G e2* $\wedge$ *tail G e1* $\neq$ *tail G e2*
proof
fix *e2*
assume *H2*: *e2* $\in$ *?E*
show *e1* $\neq$ *e2* $\longrightarrow$ *head G e1* $\neq$ *head G e2* $\wedge$ *tail G e1* $\neq$ *tail G e2*
proof−
have *e1* $\in$ (*arcs G*) $\wedge$ ((*tail G e1*) $\in$ *X* $\wedge$ (*head G e1*) = *R* (*tail G e1*)) **using** *H1* **by** *auto*
hence *1*: *e1* $\in$ (*arcs G*) $\wedge$ (*tail G e1*) $\in$ *X* **and** *2*: *R* (*tail G e1*) = (*head G e1*) **by** *auto*
have *e2* $\in$ (*arcs G*) $\wedge$ ((*tail G e2*) $\in$ *X* $\wedge$ (*head G e2*) = *R* (*tail G e2*)) **using** *H2* **by** *auto*
hence *3*: *e2* $\in$ (*arcs G*) $\wedge$ (*tail G e2*) $\in$ *X* **and** *4*: *R* (*tail G e2*) = (*head G e2*) **by** *auto*
show *?thesis* **using** *assms*(*1*) *R 1 2 3 4 assms*(*1*) *dirBD-matching*[*of G X Y R e1 e2*] **by** *auto*
qed
qed
qed
qed
qed
next
show *tails-set G* $\{e \mid e.\ e \in arcs\ G \wedge tail\ G\ e \in X \wedge head\ G\ e = R\ (tail\ G\ e)\} = X$
using *perfect*[*of G X Y*] *assms*(*1*) *R* **by** *auto*

```
    qed thus ?thesis by auto
  qed
qed

theorem Hall-digraph:
 fixes  G :: ('a::countable, 'b::countable) pre-digraph and X:: 'a set
  assumes dir-bipartite-digraph G X Y  and ∀ i∈X. finite (neighbourhood G i)
  shows (∃ E. dirBD-perfect-matching G X Y E) ⟷
  (∀ J⊆X. finite J ⟶ (card J) ≤ card (⋃ (neighbourhood G ‘ J)))
proof
  assume hip1:  ∃ E. dirBD-perfect-matching G X Y E
  show  (∀ J⊆X. finite J ⟶ (card J) ≤ card (⋃ (neighbourhood G ‘ J)))
    using hip1 assms(1−2) marriage-necessary-graph[of G X Y] by auto
next
  assume hip2: ∀ J⊆X. finite J ⟶ card J ≤ card (⋃ (neighbourhood G ‘ J))
 show ∃ E. dirBD-perfect-matching G X Y E using assms  marriage-sufficiency-graph[of
G X Y] hip2
  proof−
    have (∀ J⊆X. finite J ⟶ (card J) ≤ card (⋃ (neighbourhood G ‘ J)))
                                                  ⟶ (∃ E. dirBD-perfect-matching G
X Y E)
      using assms marriage-sufficiency-graph[of G X Y] by auto
    thus ?thesis using hip2 by auto
  qed
qed

locale set-family =
  fixes I :: 'a set and X :: 'a ⇒ 'b set

locale sdr = set-family +
  fixes repr :: 'a ⇒ 'b
  assumes inj-repr: inj-on repr I and repr-X: x ∈ I ⟹ repr x ∈ X x

locale bipartite-digraph =
  fixes X :: 'a set and  Y :: 'b set and E :: ('a × 'b) set
  assumes E-subset: E ⊆ X × Y
```

locale *Count-Nbhdfin-bipartite-digraph* =
  fixes X :: *'a:: countable set* **and** Y :: *'b:: countable set*
      and E :: *('a × 'b) set*
  assumes *E-subset*: $E \subseteq X \times Y$

  assumes *Nbhd-Tail-finite*: $\forall x \in X.$ *finite* $\{y.\ (x, y) \in E\}$

locale *matching* = *bipartite-digraph* +
  fixes M :: *('a × 'b) set*
  assumes *M-subset*: $M \subseteq E$
  assumes *M-right-unique*: $(x, y) \in M \Longrightarrow (x, y') \in M \Longrightarrow y = y'$
  assumes *M-left-unique*: $(x, y) \in M \Longrightarrow (x', y) \in M \Longrightarrow x = x'$

locale *perfect-matching* = *matching* +
  assumes *M-perfect*: *fst* ‘ $M = X$

lemma (**in** *sdr*) *perfect-matching*:
    perfect-matching I ($\bigcup i{\in}I.\ X\ i$) (*Sigma* I X) $\{(x,$ *repr* $x) | x.\ x \in I\}$
 by *unfold-locales* (*use inj-repr repr-X* **in** ‹*force simp*: *inj-on-def*›)+

lemma (**in** *perfect-matching*) *sdr*: *sdr* X ($\lambda x.\ \{y.\ (x,y) \in E\}$) ($\lambda x.$ *the-elem* $\{y.\ (x,y) \in M\}$)
proof *unfold-locales*
  define Y **where** $Y = (\lambda x.\ \{y.\ (x,y) \in M\})$
  have Y: $\exists y.\ Y\ x = \{y\}$ **if** $x \in X$ **for** x
    using *that M-right-unique M-perfect* **unfolding** *Y-def* **by** *fastforce*
  show *inj-on* ($\lambda x.$ *the-elem* (Y x)) X
    unfolding *Y-def inj-on-def*
    by (*metis* (*mono-tags*, *lifting*) *M-left-unique Y Y-def mem-Collect-eq singletonI the-elem-eq*)
  show *the-elem* (Y x) $\in \{y.\ (x, y) \in E\}$ **if** $x \in X$ **for** x
    using *Y M-subset Y-def* ‹$x \in X$› **by** *fastforce*
qed

From these transformations, the formalization of the countable version of Hall's Theorem for Graphs (more specifically, its sufficiency) can be stated as below; in words "if for any finite $X_s \subseteq X$ the subgraph induced by X_s has a perfect matching then the whole graph has a perfect matching"

theorem (**in** *Count-Nbhdfin-bipartite-digraph*) *Hall-Graph*:
 assumes $\exists g.$ *enumeration* (g:: *nat* ⇒'*a*) **and** $\exists h.$ *enumeration* (h:: *nat* ⇒'*b*)
 shows (∀ *Xs* ⊆ X. (*finite Xs*) ⟶
      (∃ *Ms*. *perfect-matching Xs*

$\{y.\ x \in Xs\ \wedge (x,y) \in E\}$
$\{(x,y).\ x \in Xs\ \wedge (x,y) \in E\}$
$Ms))$
$\longrightarrow (\exists\ M.\ \ perfect\text{-}matching\ X\ Y\ E\ M)$
proof(*unfold-locales*, *rule impI*)
assume *premisse1*: $(\forall\ Xs \subseteq X.\ (finite\ Xs) \longrightarrow$
$(\exists\ Ms.\ \ perfect\text{-}matching\ Xs$
$\{y.\ x \in Xs\ \wedge (x,y) \in E\}$
$\{(x,y).\ x \in Xs\ \wedge (x,y) \in E\}$
$Ms))$
show $(\exists\ M.\ perfect\text{-}matching\ X\ Y\ E\ M)$
proof−
have *A*: $\forall\, Xs{\subseteq}X.\ finite\ Xs \longrightarrow\ card\ Xs \leq card\ (\bigcup\ (\ (\lambda x.\ \{y.\ (x,y) \in E\})\ `\ Xs))$
proof(*rule allI*, *rule impI*)
fix *Xs*
define *Ys* **where** $Ys = \ \{y.\ x \in Xs\ \wedge (x,y) \in E\}$
define *Es* **where** $Es = \{(x,y).\ x \in Xs\ \wedge (x,y) \in E\}$
assume *hip1*: $Xs \subseteq X$
show $finite\ Xs \longrightarrow\ card\ Xs \leq card\ (\bigcup\ (\ (\lambda x.\ \{y.\ (x,y) \in E\})\ `\ Xs))$
proof
assume *hip2*: *finite Xs*
show $card\ Xs \leq card\ (\bigcup\ (\ (\lambda x.\ \{y.\ (x,y) \in E\})\ `\ Xs))$
proof−
have $(\exists\ Ms.\ perfect\text{-}matching\ Xs\ Ys\ Es\ Ms)$
using *hip1 hip2 premisse1 Ys-def Es-def* **by** *auto*
then obtain *Ms* **where** *Ms*: *perfect-matching Xs Ys Es Ms*
using *Ys-def Es-def* **by** *auto*
have *sdrXs* : $sdr\ Xs\ (\lambda x.\ \{y.\ (x,y) \in Es\})\ (\lambda x.\ the\text{-}elem\ \{y.\ (x,y) \in Ms\})$
using *Ms perfect-matching.sdr*[*of Xs Ys Es Ms*] **by** *blast*
define *Rs* **where** $Rs = \ (\lambda x.\ the\text{-}elem\ \{y.\ (x,y) \in Ms\})$
have *inj-Rs*: *inj-on Rs Xs*
using *sdrXs Rs-def sdr.inj-repr*[*of Xs* $(\lambda x.\ \{y.\ (x,y) \in Es\})$ *Rs*] **by** *auto*
have *B*: $\forall\, x.\ x{\in}Xs\ \longrightarrow\ Rs\ x \in\ (\lambda x.\ \{y.\ (x,y) \in Es\})\ x$
proof(*rule allI*, *rule impI*)
fix *x*
assume $x{\in}Xs$
thus $Rs\ x \in\ (\lambda x.\ \{y.\ (x,y) \in Es\})\ x$
using *sdrXs Rs-def sdr.repr-X*[*of Xs* $(\lambda x.\ \{y.\ (x,y) \in Es\})$ *Rs x*]
by *auto*
qed
have *YsE* : $Ys = (\bigcup x{\in}Xs.\ \{y.\ (x,\ y) \in E\})$
proof
show $Ys \subseteq (\bigcup x{\in}Xs.\ \{y.\ (x,\ y) \in E\})$
proof fix *x*
assume $x \in Ys$
thus $x \in (\bigcup x{\in}Xs.\ \{y.\ (x,\ y) \in E\})$ **using** *Ys-def* **by** *blast*
qed
next

show ($\bigcup x\in Xs.\ \{y.\ (x,\ y) \in E\}) \subseteq\ Ys$
proof fix x
assume $x \in (\bigcup x\in Xs.\ \{y.\ (x,\ y) \in E\})$
thus $x \in\ Ys$
using *Es-def Ms UN-iff bipartite-digraph.E-subset*
case-prodI matching-def mem-Collect-eq mem-Sigma-iff
perfect-matching-def **by** *fastforce*
qed
qed
have *YsFin*: *finite Ys*
using *Nbhd-Tail-finite Ys-def hip1 hip2* **by** *fastforce*
have $(\forall x\in Xs.\ Rs\ x \in (\lambda x.\ \{y.\ (x,y) \in Es\})\ x) \wedge$ *inj-on Rs Xs*
using *B inj-Rs* **by** *auto*
thus *?thesis* **using** *YsFin YsE Es-def card-inj-on-le*[*of Rs Xs Ys*] **by** *blast*
qed
qed
qed
have *premisse2*: *Count-Nbhdfin-bipartite-digraph X Y E*
by (*simp add*: *Count-Nbhdfin-bipartite-digraph-axioms*)
have *X-countable* : *countable X* **by** *simp*
have *P2*: $\exists R.$ *system-representatives* $(\lambda x.\ \{y.\ (x,y) \in E\})\ X\ R$
using *premisse2 A Hall*[*of X* $(\lambda x.\ \{y.\ (x,y) \in E\})$]
Nbhd-Tail-finite **by** *blast*
then obtain R **where** *system-representatives* $(\lambda x.\ \{y.\ (x,\ y) \in E\})\ X\ R$ **by** *auto*
hence *sdr* $X\ (\lambda x.\ \{y.\ (x,y) \in E\})\ R$ **unfolding** *system-representatives-def sdr-def* **by** *auto*
hence $\exists M.$ *perfect-matching* $X\ (\bigcup i\in X.\ (\lambda x.\ \{y.\ (x,y) \in E\})\ i)$ (*Sigma* $X\ (\lambda x.\ \{y.\ (x,y) \in E\}))\ M$
using *sdr.perfect-matching*[*of X* $(\lambda x.\ \{y.\ (x,y) \in E\})\ R$] **by** *auto*
then obtain M
where *PM0*: *perfect-matching* $X\ (\bigcup i\in X.\ (\lambda x.\ \{y.\ (x,y) \in E\})\ i)$
(*Sigma* $X\ (\lambda x.\ \{y.\ (x,y) \in E\}))\ M$ **by** *auto*
have *Ed2*: $E =$ (*Sigma* $X\ (\lambda x.\ \{y.\ (x,y) \in E\}))$
proof
show $E \subseteq$ (*SIGMA* $x{:}X.\ \{y.\ (x,\ y) \in E\})$
proof fix x
assume $x \in E$
thus $x \in$ (*SIGMA* $x{:}X.\ \{y.\ (x,\ y) \in E\})$
using *E-subset* **by** *blast*
qed
next
show (*SIGMA* $x{:}X.\ \{y.\ (x,\ y) \in E\}) \subseteq E$
proof fix x
assume $x \in$ (*SIGMA* $x{:}X.\ \{y.\ (x,\ y) \in E\})$
thus $x \in E$ **by** *blast*
qed
qed
have *PM1*: *perfect-matching* $X\ (\bigcup i\in X.\ (\lambda x.\ \{y.\ (x,y) \in E\})\ i)\ E\ M$

using *PM0 Ed2* **by** *auto*
hence *PM2*: *perfect-matching X Y E M*
using *Count-Nbhdfin-bipartite-digraph-axioms* **unfolding** *matching-def perfect-matching-def*
proof −
assume (*bipartite-digraph X* ($\bigcup$*i*∈*X*. {*y*. (*i*, *y*) ∈ *E*}) *E* ∧ *matching-axioms E M*) ∧ *perfect-matching-axioms X M*
then show (*bipartite-digraph X Y E* ∧ *matching-axioms E M*) ∧ *perfect-matching-axioms X M*
using *E-subset bipartite-digraph.intro* **by** *blast*
qed
thus *PM* : ∃*M*. *perfect-matching X Y E M* **using** *PM2* **by** *auto*
qed
qed

end

References

[1] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, second edition, 1996.

[2] F. F. Serrano Suárez. *Formalización en Isar de la Meta-Lógica de Primer Orden.* PhD thesis, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla, Spain, 2012. https://idus.us.es/handle/11441/57780. In Spanish.

[3] R. M. Smullyan. *First-Order Logic*, volume 43 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 2. Folge.* Springer-Verlag, Berlin, 1968. Also available as a Dover Publications Inc., 1994.

[4] F. F. S. Suárez, M. Ayala-Rincón, and T. A. de Lima. Hall's Theorem for Enumerable Families of Finite Sets. In *Proceedings 15th International Conference on Intelligent Computer Mathematics, CICM*, volume 13467 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2022.

[5] F. F. S. Suárez, M. Ayala-Rincón, and T. A. de Lima. Formalisation of Hall's Theorem for Countable Infinite Graphs. In *Proceedings 18th Colombian Conference on Computing, CCC.* Springer, 2024.