Hall's Marriage Theorem

Dongchen Jiang and Tobias Nipkow

March 17, 2025

Abstract

A proof of Hall's Marriage Theorem due to Halmos and Vaughan [1].

theory *Marriage*
imports *Main*
begin

theorem *marriage-necessary*:
  fixes $A :: 'a \Rightarrow 'b\ set$ **and** $I :: 'a\ set$
  assumes *finite* I **and** $\forall\ i{\in}I.\ finite\ (A\ i)$
  and $\exists R.\ (\forall i{\in}I.\ R\ i \in A\ i) \wedge inj\text{-}on\ R\ I$ (**is** $\exists R.\ ?R\ R\ A\ \&\ ?inj\ R\ A$)
  shows $\forall J{\subseteq}I.\ card\ J \leq card\ (\bigcup (A\ `\ J))$
proof *clarify*
  fix J
  assume $J \subseteq I$
  show $card\ J \leq card\ (\bigcup (A\ `\ J))$
  proof−
    from *assms*(3) **obtain** R **where** $?R\ R\ A$ **and** $?inj\ R\ A$ **by** *auto*
    have $inj\text{-}on\ R\ J$ **by**(*rule subset-inj-on*[*OF* ‹?inj R A› ‹J⊆I›])
    moreover have $(R\ `\ J) \subseteq (\bigcup (A\ `\ J))$ **using** ‹J⊆I› ‹?R R A› **by** *auto*
    moreover have *finite* $(\bigcup (A\ `\ J))$ **using** ‹J⊆I› *assms*
      by (*metis finite-UN-I finite-subset subsetD*)
    ultimately show *?thesis* **by** (*rule card-inj-on-le*)
  qed
qed

The proof by Halmos and Vaughan:

theorem *marriage-HV*:
  fixes $A :: 'a \Rightarrow 'b\ set$ **and** $I :: 'a\ set$
  assumes *finite* I **and** $\forall\ i{\in}I.\ finite\ (A\ i)$
  and $\forall J{\subseteq}I.\ card\ J \leq card\ (\bigcup (A\ `\ J))$ (**is** $?M\ A\ I$)
  shows $\exists R.\ (\forall i{\in}I.\ R\ i \in A\ i) \wedge inj\text{-}on\ R\ I$
      (**is** $?SDR\ A\ I$ **is** $\exists R.\ ?R\ R\ A\ I\ \&\ ?inj\ R\ A\ I$)
proof−
  { **fix** I
    have $finite\ I \Longrightarrow \forall i{\in}I.\ finite\ (A\ i) \Longrightarrow ?M\ A\ I \Longrightarrow ?SDR\ A\ I$

```
proof(induct arbitrary: A rule: finite-psubset-induct)
  case (psubset I)
  show ?case
  proof (cases)
    assume I={} then show ?thesis by simp
  next
    assume I ≠ {}
    have ∀ i∈I. A i ≠ {}
    proof (rule ccontr)
      assume ¬ (∀ i∈I. A i≠{})
      then obtain i where i∈I A i = {} by blast
      hence {i}⊆ I by auto
      from mp[OF spec[OF psubset.prems(2)] this] ‹A i={}›
      show False by simp
    qed
    show ?thesis
    proof cases
      assume case1: ∀ K⊂I. K≠{} ⟶ card (⋃(A ‘ K)) ≥ card K + 1
      show ?thesis
      proof−
        from ‹I≠{}› obtain n where n∈I by auto
        with ‹∀ i∈I. A i ≠ {}› have A n ≠ {} by auto
        then obtain x where x ∈ A n by auto
        let ?A′ = λi. A i − {x} let ?I′ = I − {n}
        from ‹n∈I› have ?I′ ⊂ I
          by (metis DiffD2 Diff-subset insertI1 psubset-eq)
        have fin′: ∀ i∈?I′. finite (?A′ i) using psubset.prems(1) by auto
        have ?M ?A′ ?I′
        proof clarify
          fix J
          assume J ⊆ ?I′
          hence J ⊂ I by (metis ‹I − {n} ⊂ I› subset-psubset-trans)
          show card J ≤ card (⋃i∈J. A i − {x})
          proof cases
            assume J = {} thus ?thesis by auto
          next
            assume J ≠ {}
            hence card J + 1 ≤ card(⋃(A ‘ J)) using case1 ‹J⊂I› by blast
            moreover
            have card(⋃(A ‘ J)) − 1 ≤ card (⋃i∈J. A i − {x}) (is ?l ≤ ?r)
            proof−
              have finite J using ‹J ⊂ I› psubset(1)
                by (metis psubset-imp-subset finite-subset)
              hence 1: finite(⋃(A ‘ J))
                using ‹∀ i∈I. finite(A i)› ‹J⊂I› by force
              have ?l = card(⋃(A ‘ J)) − card{x} by simp
              also have ... ≤ card(⋃(A ‘ J) − {x}) using 1
                by (metis diff-card-le-card-Diff finite.intros)
              also have ⋃(A ‘ J) − {x} = (⋃i∈J. A i − {x}) by blast
```

```
        finally show ?thesis .
      qed
      ultimately show ?thesis by arith
    qed
  qed
  from psubset(2)[OF ‹?I'⊂I› fin' ‹?M ?A' ?I'›]
  obtain R' where ?R R' ?A' ?I' ?inj R' ?A' ?I' by auto
  let ?Rx = R'(n := x)
  have ?R ?Rx A I using ‹x∈A n› ‹?R R' ?A' ?I'› by force
  have ∀ i∈?I'. ?Rx i ≠ x using ‹?R R' ?A' ?I'› by auto
  hence ?inj ?Rx A I using ‹?inj R' ?A' ?I'›
    by(auto simp: inj-on-def)
  with ‹?R ?Rx A I› show ?thesis by auto
 qed
next
 assume ¬ (∀ K⊂I. K≠{} ⟶ card (⋃(A ‘ K)) ≥ card K + 1)
 then obtain K where
   K⊂I K≠{} and c1: ¬(card (⋃(A ‘ K)) ≥ card K + 1) by auto
 with psubset.prems(2) have card (⋃(A ‘ K)) ≥ card K by auto
 with c1 have case2: card (⋃(A ‘ K))= card K by auto
 from ‹K⊂I› ‹finite I› have finite K by (auto intro:finite-subset)
 from psubset.prems ‹K⊂I›
 have ∀ i∈K. finite (A i) ∀ J⊆K. card J ≤ card(⋃(A ‘ J)) by auto
 from psubset(2)[OF ‹K⊂I› this]
 obtain R1 where ?R R1 A K ?inj R1 A K by auto
 let ?AK = λi. A i − ⋃(A ‘ K) let ?IK = I − K
 from ‹K≠{}› ‹K⊂I› have ?IK⊂I by auto
 have ∀ i∈?IK. finite (?AK i) using psubset.prems(1) by auto
 have ?M ?AK ?IK
 proof clarify
  fix J assume J ⊆ ?IK
  with ‹finite I› have finite J by(auto intro: finite-subset)
  show card J ≤ card (⋃ (?AK ‘ J))
  proof−
   from ‹J⊆?IK› have J ∩ K = {} by auto
   have card J = card(J∪K) − card K
     using ‹finite J› ‹finite K› ‹J∩K={}›
     by (auto simp: card-Un-disjoint)
   also have card(J∪K) ≤ card(⋃(A ‘ (J∪K)))
   proof −
     from ‹J⊆?IK› ‹K⊂I› have J ∪ K ⊆ I by auto
     with psubset.prems(2) show ?thesis by blast
   qed
   also have ... − card K = card(⋃ (?AK ‘ J) ∪ ⋃(A ‘ K)) − card K
   proof−
     have ⋃(A ‘ (J∪K)) = ⋃ (?AK ‘ J) ∪ ⋃(A ‘ K)
       using ‹J⊆?IK› by auto
     thus ?thesis by simp
   qed
```

```
          also have ... = card (⋃ (?AK ‘ J)) + card(⋃(A ‘ K)) − card K
          proof−
            have finite (⋃ (?AK ‘ J)) using ‹finite J› ‹J⊆?IK› psubset(3)
              by(blast intro: finite-UN-I finite-Diff)
            moreover have finite (⋃(A ‘ K))
              using ‹finite K› ‹∀ i∈K. finite (A i)› by auto
            moreover have ⋃ (?AK ‘ J) ∩ ⋃(A ‘ K) = {} by auto
            ultimately show ?thesis
              by (simp add: card-Un-disjoint del:Un-Diff-cancel2)
          qed
          also have ... = card (⋃ (?AK ‘ J)) using case2 by simp
          finally show ?thesis by simp
        qed
      qed
       from psubset(2)[OF ‹?IK⊂I› ‹∀ i∈?IK. finite (?AK i)› ‹∀ J⊆?IK. card
J ≤ card (⋃i∈J. A i − ⋃ (A ‘ K))›]
      obtain R2 where ?R R2 ?AK ?IK ?inj R2 ?AK ?IK by auto
      let ?R12 = λi. if i∈K then R1 i else R2 i
      have ∀ i∈I. ?R12 i ∈ A i using ‹?R R1 A K›‹?R R2 ?AK ?IK› by auto
      moreover have ∀ i∈I. ∀ j∈I. i≠j⟶?R12 i ≠ ?R12 j
      proof clarify
        fix i j assume i∈I j∈I i≠j ?R12 i = ?R12 j
        show False
        proof−
          { assume i∈K ∧ j∈K ∨ i∉K∧j∉K
           with ‹?inj R1 A K› ‹?inj R2 ?AK ?IK› ‹?R12 i=?R12 j› ‹i≠j› ‹i∈I›
‹j∈I›
            have ?thesis by (fastforce simp: inj-on-def)
          } moreover
          { assume i∈K ∧ j∉K ∨ i∉K ∧ j∈K
            with ‹?R R1 A K› ‹?R R2 ?AK ?IK› ‹?R12 i=?R12 j› ‹j∈I› ‹i∈I›
            have ?thesis by auto (metis Diff-iff)
          } ultimately show ?thesis by blast
        qed
      qed
      ultimately show ?thesis unfolding inj-on-def by fast
    qed
  qed
 qed
}
with assms ‹?M A I› show ?thesis by auto
qed
```

The proof by Rado:

```
theorem marriage-Rado:
  fixes A :: 'a ⇒ 'b set and I :: 'a set
  assumes finite I and ∀ i∈I. finite (A i)
  and ∀ J⊆I. card J ≤ card (⋃(A ‘ J)) (is ?M A)
  shows ∃ R. (∀ i∈I. R i ∈ A i) ∧ inj-on R I
```

```
      (is ?SDR A is ∃ R. ?R R A & ?inj R A)
proof−
 { have ∀ i∈I. finite (A i) ⟹ ?M A ⟹ ?SDR A
   proof(induct n == ∑ i∈I. card(A i) − 1 arbitrary: A)
     case 0
     have ∀ i∈I.∃ a. A(i) = {a}
     proof (rule ccontr)
       assume ¬ (∀ i∈I.∃ a. A i = {a})
       then obtain i where i: i:I ∀ a. A i ≠ {a} by blast
       hence {i}⊆ I by auto
       from 0(1−2) mp[OF spec[OF 0.prems(2)] ‹{i}⊆I›] ‹finite I› i
       show False by (auto simp: card-le-Suc-iff)
     qed
     then obtain R where R: ∀ i∈I. A i = {R i} by metis
     then have ∀ i∈I. R i ∈ A i by blast
     moreover have inj-on R I
     proof (auto simp: inj-on-def)
       fix x y assume x ∈ I y ∈ I R x = R y
       with R spec[OF 0.prems(2), of {x,y}] show x=y
         by (simp add:le-Suc-eq card-insert-if split: if-splits)
     qed
     ultimately show ?case by blast
   next
     case (Suc n)
     from Suc.hyps(2)[symmetric, THEN sum-SucD]
     obtain i where i: i:I 2 ≤ card(A i) by auto
     then obtain x1 x2 where x1 : A i x2 : A i x1 ≠ x2
       using Suc(3) by (fastforce simp: card-le-Suc-iff eval-nat-numeral)
     let ?Ai x = A i − {x} let ?A x = A(i:=?Ai x)
     let ?U J = ⋃(A ‘ J) let ?Ui J x = ?U J ∪ ?Ai x
     have n1: n = (∑ j∈I. card (?A x1 j) − 1)
       using Suc.hyps(2) Suc.prems(1) i ‹finite I› ‹x1:A i›
       by (auto simp: sum.remove card-Diff-singleton)
     have n2: n = (∑ j∈I. card (?A x2 j) − 1)
       using Suc.hyps(2) Suc.prems(1) i ‹finite I› ‹x2:A i›
       by (auto simp: sum.remove card-Diff-singleton)
     have finx1: ∀ j∈I. finite (?A x1 j) by (simp add: Suc(3))
     have finx2: ∀ j∈I. finite (?A x2 j) by (simp add: Suc(3))
     { fix x assume ¬ ?M (A(i:= ?Ai x))
       with Suc.prems(2) obtain J
         where J: J ⊆ I card J > card(⋃((A(i:= ?Ai x) ‘ J)))
         by (auto simp add:not-less-eq-eq Suc-le-eq)
       note fJi = finite-Diff[OF finite-subset[OF ‹J⊆I› ‹finite I›], of {i}]
       have fU: finite(?U (J−{i})) using ‹J⊆I›
         by (metis Diff-iff Suc(3) finite-UN[OF fJi] subsetD)
       have i ∈ J using J Suc.prems(2)
           by (simp-all add: UNION-fun-upd not-le[symmetric] del: fun-upd-apply
split: if-splits)
       hence card(J−{i}) ≥ card(?Ui (J−{i}) x)
```

using *fJi J* **by**(*simp add*: *UNION-fun-upd del*: *fun-upd-apply*)
hence $\exists\, J \subseteq I.\ i \notin J \wedge card(J) \geq card(?Ui\ J\ x) \wedge finite(?U\ J)$
by (*metis DiffD2 J*(*1*) *fU* ‹$i \in J$› *insertI1 subset-insertI2 subset-insert-iff*)
} **note** *lem* = *this*
have *?M* (*?A x1*) ∨ *?M* (*?A x2*) — Rado's Lemma
proof(*rule ccontr*)
assume ¬ (*?M* (*?A x1*) ∨ *?M* (*?A x2*))
with *lem* **obtain** *J1 J2* **where**
J1: $J1 \subseteq I$ $i \notin J1$ *card J1* ≥ *card*(*?Ui J1 x1*) *finite*(*?U J1*) **and**
J2: $J2 \subseteq I$ $i \notin J2$ *card J2* ≥ *card*(*?Ui J2 x2*) *finite*(*?U J2*)
by *metis*
note *fin1* = *finite-subset*[*OF* ‹$J1 \subseteq I$› *assms*(*1*)]
note *fin2* = *finite-subset*[*OF* ‹$J2 \subseteq I$› *assms*(*1*)]
have *finUi1*: *finite*(*?Ui J1 x1*) **using** *Suc*(*3*) **by**(*blast intro*: *J1*(*4*) *i*(*1*))
have *finUi2*: *finite*(*?Ui J2 x2*) **using** *Suc*(*3*) **by**(*blast intro*: *J2*(*4*) *i*(*1*))
have *card J1* + *card J2* + *1* = *card*($J1 \cup J2$) + *1* + *card*($J1 \cap J2$)
by *simp* (*metis card-Un-Int fin1 fin2*)
also have *card*($J1 \cup J2$) + *1* = *card*(*insert i* ($J1 \cup J2$))
using ‹$i \notin J1$› ‹$i \notin J2$› *fin1 fin2* **by** *simp*
also have ... ≤ *card* ($\bigcup$ (*A* ‘ *insert i* ($J1 \cup J2$))) (**is** - ≤ *card ?M*)
by (*metis J1*(*1*) *J2*(*1*) *Suc*(*4*) *Un-least i*(*1*) *insert-subset*)
also have *?M* = *?Ui J1 x1* ∪ *?Ui J2 x2* **using** ‹$x1 \neq x2$› **by** *auto*
also have *card*($J1 \cap J2$) ≤ *card*($\bigcup$(*A* ‘ ($J1 \cap J2$)))
by (*metis J2*(*1*) *Suc*(*4*) *le-infI2*)
also have ... ≤ *card*(*?U J1* ∩ *?U J2*) **by**(*blast intro*: *card-mono J1*(*4*))
also have ... ≤ *card*(*?Ui J1 x1* ∩ *?Ui J2 x2*)
using *Suc*(*3*) ‹$i \in I$› **by**(*blast intro*: *card-mono J1*(*4*))
finally show *False* **using** *J1*(*3*) *J2*(*3*)
by(*auto simp add*: *card-Un-Int*[*symmetric*, *OF finUi1 finUi2*])
qed
thus *?case* **using** *Suc.hyps*(*1*)[*OF n1 finx1*] *Suc.hyps*(*1*)[*OF n2 finx2*]
by (*metis DiffD1 fun-upd-def*)
qed
} **with** *assms* ‹*?M A*› **show** *?thesis* **by** *auto*
qed

end

References

[1] P. R. Halmos and H. E. Vaughan. The marriage problem. *American Journal of Mathematics*, 72:214–215, 1950.