Bondy's Theorem

Jeremy Avigad and Stefan Hetzl

March 17, 2025

Abstract

A proof of Bondy's Theorem following Bollobás [1].

theory *Bondy*
imports *Main*
begin

lemma *card-less-if-surj-not-inj*:
  ⟦ *finite A*; *f ‘ A = B*; ¬ *inj-on f A* ⟧ ⟹ *card B* < *card A*
by (*metis card-image-le inj-on-iff-eq-card order-le-neq-trans*)

theorem *Bondy* :
  assumes ∀ *A* ∈ *F*. *A* ⊆ *X* **and** *card X* ≥ *1* **and** *card F* = *card X*
  shows ∃ *D*. *D* ⊆ *X* & *card D* < *card X* & *card* (*inter D ‘ F*) = *card F*
proof −
  from *assms*(*2*,*3*) **have** *finite F* **and** *finite X*
    by (*metis card.infinite not-one-le-zero*)+
  { fix *m*
    have *m* < *card F* ⟹ ∃ *D*. *D* ⊆ *X* & *card D* ≤ *m* & *card* (*inter D ‘ F*) ≥ *m*
+ *1*
    proof (*induction m*)
      case *0*
      hence {} ⊆ *X* & *card* {} ≤ *0* & *card* (*inter* {} *‘ F*) ≥ *0* + *1*
        by *auto* (*metis Suc-leI card-eq-0-iff empty-is-image finite-imageI gr0I*)
      thus ∃ *D*. (*D* ⊆ *X* & *card D* ≤ *0* & *card* (*inter D ‘ F*) ≥ *0* + *1*) **by** *blast*
    next
      case (*Suc m*)
      hence *m* < *card F* **by** *arith*
      with *Suc.IH* **obtain** *D*
        where *D*: *D* ⊆ *X* ∧ *card D* ≤ *m* ∧ *m* + *1* ≤ *card* (*inter D ‘ F*) **by** *auto*
      with ‹*finite X*› **have** *finite D* **by** (*auto intro*: *finite-subset*)
      show *?case*
      proof (*cases card* (*inter D ‘ F*) = *card F*)
        case *True*
        hence *D* ⊆ *X* ∧ *card D* ≤ *Suc m* ∧ *Suc m* + *1* ≤ *card*(*inter D ‘ F*)
          using *D Suc.prems* **by** *auto*
        thus *?thesis* **by** *blast*
      next

```
      case False
      hence ~ inj-on (inter D) F by (auto simp: card-image)
      then obtain A1 A2 where A1 ∈ F and A2 ∈ F and
        D ∩ A1 = D ∩ A2 and A1 ≠ A2  by (auto simp: inj-on-def)
      then obtain x where x: x : (A1 − A2) ∪ (A2 − A1) by auto
      from ‹∀ A ∈ F. A ⊆ X› ‹A1 ∈ F› ‹A2 ∈ F› x have x : X by auto
      let ?E = insert x D
      from D ‹finite D› have card ?E ≤ Suc m
        by (metis (full-types) Suc-le-mono card-insert-if le-Suc-eq)
      moreover with D ‹x:X› have ?E ⊆ X by auto
      moreover have Suc m < card (inter ?E ‘ F)
      proof −
       from ‹D ∩ A1 = D ∩ A2› have 1: (D ∩ (?E ∩ A1)) = (D ∩ (?E ∩ A2))
          by auto
        from x have 2: ?E Int A1 ≠ ?E Int A2 by auto
        have 3: inter D ∘ inter ?E = inter D by auto
        have 4: ~ inj-on (inter D) (inter ?E ‘ F)
          unfolding inj-on-def using 1 2 ‹A1 ∈ F› ‹A2 ∈ F› by blast
        from D have Suc m ≤ card (inter D ‘ F) by auto
        also have ... < card (inter ?E ‘ F)
          by (rule card-less-if-surj-not-inj[of - inter D])
            (auto simp add: image-image 3 4 ‹finite F›)
        finally show ?thesis .
      qed
       ultimately have ?E⊆X ∧ card ?E ≤ Suc m ∧ Suc m + 1 ≤ card (inter
?E ‘ F)
        by auto
      thus ∃ D⊆X. card D ≤ Suc m ∧ Suc m + 1 ≤ card (inter D ‘ F) by blast
    qed
  qed
 }
 moreover from assms(2,3) have card X − 1 < card F by auto
 ultimately obtain D where
   D ⊆ X & card D ≤ card X − 1 & card (inter D ‘ F) ≥ (card X − 1) + 1
   by auto
 moreover with ‹finite F› have card (inter D ‘ F) ≤ card F
   by (elim card-image-le)
 ultimately have D ⊆ X & card D < card X & card (inter D ‘ F) = card F
   using ‹card F = card X› by auto
 thus ?thesis by auto
qed

end
```

References

[1] B. Bollobás. *Combinatorics: set systems, hypergraphs, families of vectors and combinatorial probability.* Cambridge University Press, 1986.