Chvátal's Theorem on the Ramsey Number of a Triangle versus a Tree

Carles Marín

November 11, 2026

Abstract

We formalize Chvátal's theorem [1]: for every tree T with n vertices, the graph Ramsey number $R(K_3, T)$ equals $2n-1$. The upper bound is proved by induction on n, removing a leaf; the lower bound is the complete bipartite graph with parts of sizes $n-1$ and $n-1$. As a corollary, every connected graph with n vertices and at most $n-1$ edges, being a tree, has the same Ramsey number. Graphs, cliques and Ramsey numbers are those of the entry *Ramsey_Bounds*, and trees those of the entry *Tree_Enumeration*.

Acknowledgement. The formalization and the text of this document were prepared with the assistance of generative AI systems, and checked by the author.

Contents

1 Chvátal's theorem: the Ramsey number of a triangle versus a tree **2**

2 The upper bound **2**

3 The lower bound **4**

4 Chvátal's theorem **5**

5 Agreement with *RN* of *Ramsey-Bounds* **6**

theory *Chvatal-Tree-Ramsey*
imports *Ramsey-Bounds.Ramsey-Bounds Tree-Enumeration.Tree-Graph*
begin

1 Chvátal's theorem: the Ramsey number of a triangle versus a tree

The host graphs and Ramsey numbers follow the AFP entry *Ramsey-Bounds*: a graph on a finite set W is a set of edges, cliques and independent sets are those of *HOL−Library.Ramsey*, and the definition below has the shape of *is-clique-RN*. Trees are those of the AFP entry *Tree-Enumeration* (locale *tree*: a finite connected undirected graph without cycles). A copy of a tree (V_T, E_T) in the complement of a graph is an injective map of V_T into W sending every edge of the tree to a non-edge of the graph.

definition *compl-copy* :: *'a set ⇒ 'a set set ⇒ 'b set ⇒ 'b set set ⇒ bool* **where**
compl-copy VT ET W E ⟷
(∃*f*. *inj-on f VT* ∧ *f ‘ VT* ⊆ *W* ∧ (∀*u v*. {*u*, *v*} ∈ *ET* ⟶ {*f u*, *f v*} ∉ *E*))

definition *is-triangle-tree-RN* **where**
is-triangle-tree-RN ≡ λ*U*::*'b itself*. λ*VT ET r*.
(∀ *W*::*'b set*. ∀ *E*. *finite W* ⟶ *card W* ≥ *r* ⟶
(∃ *K*⊆*W*. *card K* = *3* ∧ *clique K E*) ∨ *compl-copy VT ET W E*)

The Ramsey number $R(K_3, T)$, with vertex sets of natural numbers as in *RN*.

definition *triangle-tree-RN* :: *'a set ⇒ 'a set set ⇒ nat* **where**
triangle-tree-RN VT ET = (*LEAST r*. *is-triangle-tree-RN TYPE*(*nat*) *VT ET r*)

lemma *is-triangle-tree-RN-mono*:
assumes *is-triangle-tree-RN U VT ET r* **and** *r* ≤ *s*
shows *is-triangle-tree-RN U VT ET s*
using *assms* **unfolding** *is-triangle-tree-RN-def* **by** (*meson order-trans*)

2 The upper bound

Induction on the tree, adding a leaf l at v. Let R be the part of W not used by the copy of the smaller tree. If some vertex of R is not adjacent to the image of v, it receives l. Otherwise R lies in the neighbourhood of that image, so it is independent because there is no triangle, and it is large enough to hold the whole tree.

lemma *chvatal-upper-aux*:
fixes *W* :: *'b set* **and** *H* :: *'b set set*
assumes *tr*: *tree VT ET* **and** *finW*: *finite W*
and *noK*: ¬ (∃ *K*⊆*W*. *card K* = *3* ∧ *clique K H*) **and** *big*: *2* ∗ *card VT* − *1* ≤ *card W*
shows *compl-copy VT ET W H*
using *tr noK big*
proof (*induct rule*: *tree-induct*)
case (*singolton v*)
then obtain *w* **where** *w* ∈ *W* **by** *force*
then show *?case*
unfolding *compl-copy-def* **by** (*intro exI*[**where** *x* = λ-. *w*]) *auto*
next
case (*insert l v V′ E′*)
then interpret *t*: *tree V′ E′* **by** *auto*
have *finV*: *finite V′* **by** (*rule t.finV*)
have *c*: *card* (*insert l V′*) = *Suc* (*card V′*)
by (*simp add*: *local.insert t.finV*)
have *2* ∗ *card V′* − *1* ≤ *card W* **using** *insert c* **by** *arith*
with *insert* **have** *compl-copy V′ E′ W H* **by** *blast*
then obtain *f* **where** *fi*: *inj-on f V′* **and** *fW*: *f ‘ V′* ⊆ *W*
and *fe*: ∀ *a b*. {*a*, *b*} ∈ *E′* ⟶ {*f a*, *f b*} ∉ *H*
unfolding *compl-copy-def* **by** *blast*
let *?R* = *W* − *f ‘ V′*
have *finR*: *finite ?R* **using** *finW* **by** *simp*
have *cfV*: *card* (*f ‘ V′*) = *card V′* **by** (*rule card-image*[*OF fi*])
have *cR*: *card ?R* = *card W* − *card V′*
using *card-Diff-subset*[*OF finite-imageI*[*OF finV*] *fW*] *cfV* **by** *simp*
have *cRge*: *card* (*insert l V′*) ≤ *card ?R* **using** *cR insert c* **by** *arith*
show *compl-copy* (*insert l V′*) (*insert* {*l*, *v*} *E′*) *W H*

```
proof (cases ∃ w∈?R. {f v, w} ∉ H)
  case True
  then obtain w where wR: w ∈ ?R and nw: {f v, w} ∉ H by blast
  have nw': {w, f v} ∉ H using nw by (simp add: insert-commute)
  define g where g = (λx. if x = l then w else f x)
  have gV: ⋀x. x ∈ V' ⟹ g x = f x using insert by (auto simp: g-def)
  have gl: g l = w by (simp add: g-def)
  have giV: inj-on g V' by (rule iffD2[OF inj-on-cong[OF gV] fi])
  have wn: g l ∉ g ‘ V' using wR gl gV by force
  have VV: V' − {l} = V' using insert by blast
  have inj: inj-on g (insert l V')
    unfolding inj-on-insert VV using giV wn by blast
  have gW: g ‘ insert l V' ⊆ W
    using fW gV gl wR by auto
  have ge: {g a, g b} ∉ H if ab: {a, b} ∈ insert {l, v} E' for a b
  proof −
    show {g a, g b} ∉ H
    proof (cases {a, b} = {l, v})
      case True
      then have (a = l ∧ b = v) ∨ (a = v ∧ b = l) by (auto simp: doubleton-eq-iff)
      then show ?thesis using nw nw' gl gV insert by auto
    next
      case False
      with ab have e: {a, b} ∈ E' by blast
      have {a, b} ⊆ V' by (rule t.wellformed[OF e])
      then have a ∈ V' b ∈ V' by auto
      then show ?thesis using fe e gV by simp
    qed
  qed
  show ?thesis
    unfolding compl-copy-def by (rule exI[where x = g]) (use inj gW ge in blast)
next
  case False
  then have allH: ⋀w. w ∈ ?R ⟹ {f v, w} ∈ H by blast
  have fvW: f v ∈ W using fW insert by blast
  have fvR: f v ∉ ?R using insert by blast
  have ind: ⋀x y. x ∈ ?R ⟹ y ∈ ?R ⟹ x ≠ y ⟹ {x, y} ∉ H
  proof (rule notI)
    fix x y assume x: x ∈ ?R and y: y ∈ ?R and xy: x ≠ y and e: {x, y} ∈ H
    have nx: f v ≠ x using x fvR by blast
    have ny: f v ≠ y using y fvR by blast
    have cK: card {f v, x, y} = 3 using nx ny xy by (simp add: card-insert-if)
    have KW: {f v, x, y} ⊆ W using fvW x y by blast
    have clique {f v, x, y} H
      using allH[OF x] allH[OF y] e by (auto simp: clique-def insert-commute)
    then show False using noK cK KW by blast
  qed
  have finI: finite (insert l V') using finV by simp
  obtain h where h1: h ‘ insert l V' ⊆ ?R and h2: inj-on h (insert l V')
    using card-le-inj[OF finI finR cRge] by blast
  have he: {h a, h b} ∉ H if ab: {a, b} ∈ insert {l, v} E' for a b
  proof −
    have mem: a ∈ insert l V' ∧ b ∈ insert l V' ∧ a ≠ b
    proof (cases {a, b} = {l, v})
      case True
      then have (a = l ∧ b = v) ∨ (a = v ∧ b = l) by (auto simp: doubleton-eq-iff)
      then show ?thesis using insert by auto
    next
      case False
      with ab have e: {a, b} ∈ E' by blast
      have {a, b} ⊆ V' by (rule t.wellformed[OF e])
      moreover have card {a, b} = 2 by (rule t.two-edges[OF e])
      ultimately show ?thesis by auto
    qed
    have ha: h a ∈ ?R using mem h1 by blast
    have hb: h b ∈ ?R using mem h1 by blast
```

```
    have hn: h a ≠ h b using mem h2 by (meson inj-onD)
    show {h a, h b} ∉ H by (rule ind[OF ha hb hn])
   qed
   have hW: h ‘ insert l V' ⊆ W using h1 by blast
   show ?thesis
    unfolding compl-copy-def by (rule exI[where x = h]) (use h2 hW he in blast)
  qed
qed


theorem chvatal-upper:
  assumes tree VT ET
  shows is-triangle-tree-RN U VT ET (2 * card VT - 1)
  unfolding is-triangle-tree-RN-def using chvatal-upper-aux[OF assms] by blast
```

3 The lower bound

```
lemma tree-constant-on-edges:
  assumes tr: tree V E and h: ⋀a b. {a, b} ∈ E ⟹ Q a = Q b
   and x: x ∈ V and y: y ∈ V
  shows Q x = Q y
  using assms
proof (induction arbitrary: x y rule: tree-induct)
  case (singolton v)
  then show ?case by simp
next
  case (insert l v V E)
  then show ?case
   by (metis insert-iff)
qed
```

The complete bipartite graph with two parts of size $n-1$ has no triangle, and a connected graph in its complement stays inside one part.

```
theorem chvatal-lower:
  fixes U :: 'b itself
  assumes tr: tree VT ET and inf: infinite (UNIV :: 'b set)
  shows ¬ is-triangle-tree-RN U VT ET (2 * card VT - 2)
proof
  assume RN: is-triangle-tree-RN U VT ET (2 * card VT - 2)
  interpret t: tree VT ET by (rule tr)
  have n1: 1 ≤ card VT using t.finV t.not-empty by (simp add: Suc-le-eq card-gt-0-iff)
  obtain A :: 'b set where finA: finite A and cA: card A = card VT - 1
   using infinite-arbitrarily-large[OF inf] by blast
  have infinite (UNIV - A) by (rule Diff-infinite-finite[OF finA inf])
  then obtain B :: 'b set where finB: finite B and cB: card B = card VT - 1
   and BA: B ⊆ UNIV - A
   using infinite-arbitrarily-large by blast
  have disj: A ∩ B = {} using BA by blast
  define W where W = A ∪ B
  define H where H = {{a, b} | a b. a ∈ A ∧ b ∈ B}
  have finW: finite W using finA finB by (simp add: W-def)
  have cW: 2 * card VT - 2 ≤ card W
   unfolding W-def using card-Un-disjoint[OF finA finB disj] cA cB n1 by arith
  have sep: (x ∈ A) ≠ (y ∈ A) if {x, y} ∈ H for x y
  proof -
   from that obtain a b where e: {x, y} = {a, b} and a: a ∈ A and b: b ∈ B
    unfolding H-def by blast
   have nb: b ∉ A using disj b by blast
   show (x ∈ A) ≠ (y ∈ A) using e a nb by (auto simp: doubleton-eq-iff)
  qed
  have notri: ¬ (∃K⊆W. card K = 3 ∧ clique K H)
  proof
   assume ∃K⊆W. card K = 3 ∧ clique K H
   then obtain K where cK: card K = 3 and qK: clique K H by blast
   then obtain x y z where K: K = {x, y, z} and xy: x ≠ y and yz: y ≠ z and xz: x ≠ z
```

```
    using card-3-iff[of K] by blast
   have e1: {x, y} ∈ H and e2: {y, z} ∈ H and e3: {x, z} ∈ H
    using qK K xy yz xz by (auto simp: clique-def)
   show False using sep[OF e1] sep[OF e2] sep[OF e3] by blast
  qed
  have (∃ K⊆W. card K = 3 ∧ clique K H) ∨ compl-copy VT ET W H
   using RN finW cW unfolding is-triangle-tree-RN-def by blast
  with notri have compl-copy VT ET W H by blast
  then obtain f where fi: inj-on f VT and fW: f ‘ VT ⊆ W
   and fe: ∀ a b. {a, b} ∈ ET ⟶ {f a, f b} ∉ H
   unfolding compl-copy-def by blast
  have side: (f a ∈ A) = (f b ∈ A) if ab: {a, b} ∈ ET for a b
  proof −
   have sub: {a, b} ⊆ VT by (rule t.wellformed[OF ab])
   have fa: f a ∈ A ∪ B f b ∈ A ∪ B using fW sub by (auto simp: W-def)
   have nH: {f a, f b} ∉ H using fe ab by blast
   show (f a ∈ A) = (f b ∈ A)
    using H-def fa nH by blast
  qed
  have same: ⋀x y. x ∈ VT ⟹ y ∈ VT ⟹ (f x ∈ A) = (f y ∈ A)
   using tree-constant-on-edges[where Q = λx. f x ∈ A, OF tr side] by blast
  obtain x0 where x0: x0 ∈ VT using t.not-empty by blast
  show False
  proof (cases f x0 ∈ A)
   case True
   have f ‘ VT ⊆ A using True same x0 by blast
   from card-inj-on-le[OF fi this finA] cA n1 show False by arith
  next
   case False
   then have f ‘ VT ⊆ B
    using W-def fW same x0
    by (metis Un-iff image-subset-iff)
   from card-inj-on-le[OF fi this finB] cB n1 show False by arith
  qed
qed
```

4 Chvátal's theorem

```
theorem chvatal:
  assumes tr: tree VT ET
  shows triangle-tree-RN VT ET = 2 ∗ card VT − 1
  unfolding triangle-tree-RN-def
proof (rule Least-equality)
  show is-triangle-tree-RN TYPE(nat) VT ET (2 ∗ card VT − 1) by (rule chvatal-upper[OF tr])
next
  fix r assume r: is-triangle-tree-RN TYPE(nat) VT ET r
  interpret t: tree VT ET by (rule tr)
  have n1: 1 ≤ card VT using t.finV t.not-empty by (simp add: Suc-le-eq card-gt-0-iff)
  show 2 ∗ card VT − 1 ≤ r
  proof (rule ccontr)
   assume ¬ 2 ∗ card VT − 1 ≤ r
   then have r ≤ 2 ∗ card VT − 2 using n1 by arith
   with r have is-triangle-tree-RN TYPE(nat) VT ET (2 ∗ card VT − 2)
    by (rule is-triangle-tree-RN-mono)
   with chvatal-lower[OF tr infinite-UNIV-nat] show False by blast
  qed
qed
```

A connected graph with n vertices and at most $n - 1$ edges is a tree.

```
corollary connected-few-edges-triangle-good:
  assumes g: fin-connected-sgraph V E and c: card E + 1 ≤ card V
  shows triangle-tree-RN V E = 2 ∗ card V − 1
proof −
  interpret g: fin-connected-sgraph V E by (rule g)
  obtain T where T: spanning-tree V E T using g.has-spanning-tree by blast
  interpret s: spanning-tree V E T by (rule T)
```

have *card T* ≤ *card E* **by** (*intro card-mono g.fin-edges s.subgraph*)
moreover have *card V* = *Suc* (*card T*) **by** (*rule s.T.card-V-card-E*)
ultimately have *card V* = *Suc* (*card E*) **using** *c* **by** *arith*
then have *tree V E* **using** *card-E-treeI g.fin-connected-ulgraph-axioms* **by** *blast*
then show *?thesis* **by** (*rule chvatal*)
qed

5 Agreement with *RN* of *Ramsey-Bounds*

The tree K_2 is also a complete graph, so there the two notions of Ramsey number can be compared directly: a copy of K_2 in the complement is an independent pair.

lemma *compl-copy-edge-iff*:
fixes *W* :: *'b set* **and** *H* :: *'b set set*
assumes *uv*: *u* ≠ *v*
shows *compl-copy* {*u*, *v*} {{*u*, *v*}} *W H* ⟷ (∃ *K*⊆*W*. *card K* = *2* ∧ *indep K H*)
proof
assume *compl-copy* {*u*, *v*} {{*u*, *v*}} *W H*
then obtain *f* **where** *fi*: *inj-on f* {*u*, *v*} **and** *fW*: *f* ‘ {*u*, *v*} ⊆ *W*
and *fe*: ∀ *a b*. {*a*, *b*} ∈ {{*u*, *v*}} ⟶ {*f a*, *f b*} ∉ *H*
by (*meson compl-copy-def*)
have *ne*: *f u* ≠ *f v* **using** *fi uv* **by** (*auto simp*: *inj-on-def*)
have *nH*: {*f u*, *f v*} ∉ *H* **using** *fe* **by** *blast*
have *indep* {*f u*, *f v*} *H* **using** *nH* **by** (*auto simp*: *indep-def insert-commute*)
moreover have *card* {*f u*, *f v*} = *2* **using** *ne* **by** *simp*
moreover have {*f u*, *f v*} ⊆ *W* **using** *fW* **by** *blast*
ultimately show ∃ *K*⊆*W*. *card K* = *2* ∧ *indep K H* **by** *blast*
next
assume ∃ *K*⊆*W*. *card K* = *2* ∧ *indep K H*
then obtain *K* **where** *KW*: *K* ⊆ *W* **and** *cK*: *card K* = *2* **and** *iK*: *indep K H* **by** *blast*
then obtain *x y* **where** *K*: *K* = {*x*, *y*} **and** *xy*: *x* ≠ *y* **using** *card-2-iff*[*of K*] **by** *blast*
define *f* **where** *f* = (λ*z*. *if z* = *u then x else y*)
have *fu*: *f u* = *x* **and** *fv*: *f v* = *y* **using** *uv* **by** (*auto simp*: *f-def*)
have *fi*: *inj-on f* {*u*, *v*} **using** *fu fv xy* **by** (*auto simp*: *inj-on-def*)
have *fW*: *f* ‘ {*u*, *v*} ⊆ *W* **using** *fu fv KW K* **by** *auto*
have *nH*: {*x*, *y*} ∉ *H* {*y*, *x*} ∉ *H* **using** *iK K xy* **by** (*auto simp*: *indep-def*)
have *fe*: ∀ *a b*. {*a*, *b*} ∈ {{*u*, *v*}} ⟶ {*f a*, *f b*} ∉ *H*
using *fu fv nH* **by** (*auto simp*: *doubleton-eq-iff*)
show *compl-copy* {*u*, *v*} {{*u*, *v*}} *W H*
unfolding *compl-copy-def* **by** (*rule exI*[**where** *x* = *f*]) (*use fi fW fe* **in** *blast*)
qed

lemma *ex-clique-indep-iff*:
(∃ *K*⊆*W*. *clique-indep m n K E*) ⟷
(∃ *K*⊆*W*. *card K* = *m* ∧ *clique K E*) ∨ (∃ *K*⊆*W*. *card K* = *n* ∧ *indep K E*)
by (*simp only*: *clique-indep-def conj-disj-distribL ex-disj-distrib*)

lemma *is-triangle-tree-RN-edge-iff*:
assumes *u* ≠ *v*
shows *is-triangle-tree-RN U* {*u*, *v*} {{*u*, *v*}} *r* ⟷ *is-clique-RN U 3 2 r*
unfolding *is-triangle-tree-RN-def is-clique-RN-def compl-copy-edge-iff*[*OF assms*] *ex-clique-indep-iff*
by *simp*

The rhs is (of course) equal to 3

corollary *triangle-tree-RN-edge*:
assumes *u* ≠ *v*
shows *triangle-tree-RN* {*u*, *v*} {{*u*, *v*}} = *RN 3 2*
using *chvatal*[*OF tree2*] *assms* **by** *force*

end

References

[1] V. Chvátal. Tree-complete graph Ramsey numbers. *Journal of Graph Theory*, 1(1):93, 1977.