Clique is not solvable by monotone circuits of polynomial size*

René Thiemann
University of Innsbruck

May 8, 2022

Abstract

Given a graph G with n vertices and a number s, the decision problem Clique asks whether G contains a fully connected subgraph with s vertices. For this NP-complete problem there exists a non-trivial lower bound: no monotone circuit of a size that is polynomial in n can solve Clique.

This entry provides an Isabelle/HOL formalization of a concrete lower bound (the bound is $\sqrt[7]{n}^{\sqrt[8]{n}}$ for the fixed choice of $s = \sqrt[4]{n}$), following a proof by Gordeev.

Contents

*We thank Lev Gordeev for several clarification regarding his proof, for his explanation of the history of the underlying proof idea, and for a lively and ongoing interesting discussion on how his draft can be repaired.

1 Introduction

In this AFP submission we verify the result, that no polynomial-sized circuit can implement the Clique problem.

We arrived at this formalization by trying to verify an unpublished draft of Gordeev [4], which tries to show that Clique cannot be solved by any polynomial-sized circuit, including non-monotone ones, where the concrete exponential lower bound is $\sqrt[7]{n}^{\sqrt[8]{n}}$ for graphs with n vertices and cliques of size $s = \sqrt[4]{n}$.

Although there are some flaws in that draft, all of these disappear if one restricts to monotone circuits. Consequently, the claimed lower bound is valid for monotone circuits.

We verify a simplified version of Gordeev's proof, where those parts that deal with negations in circuits have been eliminated from definitions and proofs.

Gordeev's work itself was inspired by "Razborov's theorem" in a textbook by Papadimitriou [5], which states that Clique cannot be encoded with a monotone circuit of polynomial size. However the proof in the draft uses a construction based on the sunflower lemma of Erdős and Rado [3], following a proof in Boppana and Sipser [2]. There are further proofs on lower bounds of monotone circuits for Clique. For instance, an early result is due to Alon and Boppana [1], where they show a slightly different lower bound (using a differently structured proof without the construction based on sunflowers.)

2 Preliminaries

theory *Preliminaries*
  imports
    Main
    HOL.Real
    HOL−Library.FuncSet
begin

lemma *exists-subset-between*:
  assumes
    card A ≤ *n*
    n ≤ *card C*
    A ⊆ *C*
    finite C
  shows ∃ *B*. *A* ⊆ *B* ∧ *B* ⊆ *C* ∧ *card B* = *n*
  ⟨*proof*⟩

lemma *fact-approx-add*: *fact* (*l* + *n*) ≤ *fact l* ∗ (*real l* + *real n*) ^ *n*
⟨*proof*⟩

lemma *fact-approx-minus*: **assumes** $k \geq n$
shows $fact\ k \leq fact\ (k - n) * (real\ k\ \widehat{}\ n)$
⟨*proof*⟩

lemma *fact-approx-upper-add*: **assumes** *al*: $a \leq Suc\ l$ **shows** $fact\ l * real\ a\ \widehat{}\ n \leq fact\ (l + n)$
⟨*proof*⟩

lemma *fact-approx-upper-minus*: **assumes** $n \leq k$ **and** $n + a \leq Suc\ k$
shows $fact\ (k - n) * real\ a\ \widehat{}\ n \leq fact\ k$
⟨*proof*⟩

lemma *choose-mono*: $n \leq m \Longrightarrow n\ choose\ k \leq m\ choose\ k$
⟨*proof*⟩

lemma *div-mult-le*: $(a\ div\ b) * c \leq (a * c)\ div\ (b :: nat)$
⟨*proof*⟩

lemma *div-mult-pow-le*: $(a\ div\ b)\widehat{}n \leq a\widehat{}n\ div\ (b :: nat)\widehat{}n$
⟨*proof*⟩

lemma *choose-inj-right*:
assumes *id*: $(n\ choose\ l) = (k\ choose\ l)$
and *n0*: $n\ choose\ l \neq 0$
and *l0*: $l \neq 0$
shows $n = k$
⟨*proof*⟩

lemma *card-funcsetE*: $finite\ A \Longrightarrow card\ (A \rightarrow_E B) = card\ B\ \widehat{}\ card\ A$
⟨*proof*⟩

lemma *card-inj-on-subset-funcset*: **assumes** *finB*: *finite B*
and *finC*: *finite C*
and *AB*: $A \subseteq B$
shows $card\ \{f.\ f \in B \rightarrow_E C \wedge inj\text{-}on\ f\ A\} =$
$card\ C\widehat{}(card\ B - card\ A) * prod\ ((-)\ (card\ C))\ \{0\ ..<\ card\ A\}$
⟨*proof*⟩

end

3 Monotone Formulas

We define monotone formulas, i.e., without negation, and show that usually the constant TRUE is not required.

theory *Monotone-Formula*
imports *Main*
begin

3.1 Definition

datatype *'a mformula* =
  TRUE | *FALSE* | — True and False
  Var 'a | — propositional variables
  Conj 'a mformula 'a mformula | — conjunction
  Disj 'a mformula 'a mformula — disjunction

the set of subformulas of a mformula

fun *SUB* :: *'a mformula* ⇒ *'a mformula set* **where**
  SUB (*Conj* φ ψ) = {*Conj* φ ψ} ∪ *SUB* φ ∪ *SUB* ψ
| *SUB* (*Disj* φ ψ) = {*Disj* φ ψ} ∪ *SUB* φ ∪ *SUB* ψ
| *SUB* (*Var x*) = {*Var x*}
| *SUB FALSE* = {*FALSE*}
| *SUB TRUE* = {*TRUE*}

the variables of a mformula

fun *vars* :: *'a mformula* ⇒ *'a set* **where**
  vars (*Var x*) = {*x*}
| *vars* (*Conj* φ ψ) = *vars* φ ∪ *vars* ψ
| *vars* (*Disj* φ ψ) = *vars* φ ∪ *vars* ψ
| *vars FALSE* = {}
| *vars TRUE* = {}

lemma *finite-SUB*[*simp*, *intro*]: *finite* (*SUB* φ)
  ⟨*proof*⟩

The circuit-size of a mformula: number of subformulas

definition *cs* :: *'a mformula* ⇒ *nat* **where**
  cs φ = *card* (*SUB* φ)

variable assignments

type-synonym *'a VAS* = *'a* ⇒ *bool*

evaluation of mformulas

fun *eval* :: *'a VAS* ⇒ *'a mformula* ⇒ *bool* **where**
  eval ϑ *FALSE* = *False*
| *eval* ϑ *TRUE* = *True*
| *eval* ϑ (*Var x*) = ϑ *x*
| *eval* ϑ (*Disj* φ ψ) = (*eval* ϑ φ ∨ *eval* ϑ ψ)
| *eval* ϑ (*Conj* φ ψ) = (*eval* ϑ φ ∧ *eval* ϑ ψ)

lemma *eval-vars*: **assumes** ⋀ *x*. *x* ∈ *vars* φ ⟹ $\vartheta 1$ *x* = $\vartheta 2$ *x*
  shows *eval* $\vartheta 1$ φ = *eval* $\vartheta 2$ φ
  ⟨*proof*⟩

3.2 Conversion of mformulas to true-free mformulas

inductive-set *tf-mformula* :: *'a mformula set* **where**

tf-False: $FALSE \in$ *tf-mformula*
| *tf-Var*: $Var\ x \in$ *tf-mformula*
| *tf-Disj*: $\varphi \in$ *tf-mformula* $\Longrightarrow \psi \in$ *tf-mformula* $\Longrightarrow Disj\ \varphi\ \psi \in$ *tf-mformula*
| *tf-Conj*: $\varphi \in$ *tf-mformula* $\Longrightarrow \psi \in$ *tf-mformula* $\Longrightarrow Conj\ \varphi\ \psi \in$ *tf-mformula*

fun *to-tf-formula* **where**
to-tf-formula (*Disj phi psi*) = (*let phi′* = *to-tf-formula phi*; *psi′* = *to-tf-formula psi*
in (*if phi′* = *TRUE* ∨ *psi′* = *TRUE then TRUE else Disj phi′ psi′*))
| *to-tf-formula* (*Conj phi psi*) = (*let phi′* = *to-tf-formula phi*; *psi′* = *to-tf-formula psi*
in (*if phi′* = *TRUE then psi′ else if psi′* = *TRUE then phi′ else Conj phi′ psi′*))

| *to-tf-formula phi* = *phi*

lemma *eval-to-tf-formula*: *eval ϑ* (*to-tf-formula φ*) = *eval ϑ φ*
⟨*proof*⟩

lemma *to-tf-formula*: *to-tf-formula φ* ≠ *TRUE* ⟹ *to-tf-formula φ* ∈ *tf-mformula*

⟨*proof*⟩

lemma *vars-to-tf-formula*: *vars* (*to-tf-formula φ*) ⊆ *vars φ*
⟨*proof*⟩

lemma *SUB-to-tf-formula*: *SUB* (*to-tf-formula φ*) ⊆ *to-tf-formula* ‘ *SUB φ*
⟨*proof*⟩

lemma *cs-to-tf-formula*: *cs* (*to-tf-formula φ*) ≤ *cs φ*
⟨*proof*⟩

lemma *to-tf-mformula*: **assumes** ¬ *eval ϑ φ*
shows ∃ *ψ* ∈ *tf-mformula*. (∀ *ϑ*. *eval ϑ φ* = *eval ϑ ψ*) ∧ *vars ψ* ⊆ *vars φ* ∧ *cs ψ* ≤ *cs φ*
⟨*proof*⟩

end

4 Simplied Version of Gordeev's Proof for Monotone Circuits

4.1 Setup of Global Assumptions and Proofs of Approximations

theory *Assumptions-and-Approximations*
imports
HOL−Real-Asymp.Real-Asymp
Stirling-Formula.Stirling-Formula

Preliminaries
begin

locale *first-assumptions* =
  fixes *l p k* :: *nat*
  assumes *l2*: *l* > *2*
  and *pl*: *p* > *l*
  and *kp*: *k* > *p*
begin

lemma *k2*: *k* > *2* ⟨*proof*⟩
lemma *p*: *p* > *2* ⟨*proof*⟩
lemma *k*: *k* > *l* ⟨*proof*⟩

definition *m* = *k*^*4*

lemma *km*: *k* < *m*
  ⟨*proof*⟩

lemma *lm*: *l* + *1* < *m* ⟨*proof*⟩

lemma *m2*: *m* > *2* ⟨*proof*⟩

lemma *mp*: *m* > *p* ⟨*proof*⟩

definition *L* = *fact l* ∗ (*p* − *1*) ^ *l* + *1*

lemma *kml*: *k* ≤ *m* − *l*
⟨*proof*⟩
end

locale *second-assumptions* = *first-assumptions* +
  assumes *kl2*: *k* = *l*^*2*
  and *l8*: *l* ≥ *8*
begin

lemma *Lm*: *L* ≥ *m*
⟨*proof*⟩

lemma *Lp*: *L* > *p* ⟨*proof*⟩

lemma *L3*: *L* > *3* ⟨*proof*⟩
end

definition *eps* = *1*/(*1000* :: *real*)
lemma *eps*: *eps* > *0* ⟨*proof*⟩

definition *L0* :: *nat* **where**
  L0 = (*SOME l0*. ∀ *l*≥*l0*. *1* / *3* < (*1* + − *1* / *real l*) ^ *l*)

definition *M0* :: *nat* **where**
M0 = (*SOME y*. ∀ *x*. *x* ≥ *y* ⟶ (*root 8* (*real x*) ∗ *log 2* (*real x*) + *1*) / *real x powr* (*1* / *8* + *eps*) ≤ *1*)

definition *L0′* :: *nat* **where**
L0′ = (*SOME l0*. ∀ *n* ≥ *l0*. *6* ∗ (*real n*)^*16* ∗ *fact n* < *real* (*n*^*2* ^ *4*) *powr* (*1* / *8* ∗ *real* (*n*^*2* ^ *4*) *powr* (*1* / *8*)))

definition *L0″* :: *nat* **where** *L0″* = (*SOME l0*. ∀ *l* ≥ *l0*. *real l* ∗ *log 2* (*real* (*l*^*2* ^ *4*)) + *1* < *real* (*l*^*2*))

lemma *L0″*: **assumes** *l* ≥ *L0″* **shows** *real l* ∗ *log 2* (*real* (*l*^*2* ^ *4*)) + *1* < *real* (*l*^*2*)
⟨*proof*⟩

definition *M0′* :: *nat* **where**
M0′ = (*SOME x0*. ∀ *x* ≥ *x0*. *real x powr* (*2* / *3*) ≤ *x powr* (*3* / *4*) − *1*)

locale *third-assumptions* = *second-assumptions* +
assumes *pllog*: *l* ∗ *log 2 m* ≤ *p real p* ≤ *l* ∗ *log 2 m* + *1*
and *L0*: *l* ≥ *L0*
and *L0′*: *l* ≥ *L0′*
and *M0′*: *m* ≥ *M0′*
and *M0*: *m* ≥ *M0*
begin

lemma *approximation1*:
(*real* (*k* − *1*)) ^ (*m* − *l*) ∗ *prod* (λ *i*. *real* (*k* − *1* − *i*)) {*0*..<*l*}
> (*real* (*k* − *1*)) ^ *m* / *3*
⟨*proof*⟩

lemma *approximation2*: **fixes** *s* :: *nat*
assumes *m choose k* ≤ *s* ∗ *L*^*2* ∗ (*m* − *l* − *1 choose* (*k* − *l* − *1*))
shows ((*m* − *l*) / *k*)^*l* / (*6* ∗ *L*^*2*) < *s*
⟨*proof*⟩

lemma *approximation3*: **fixes** *s* :: *nat*
assumes (*k* − *1*)^*m* / *3* < (*s* ∗ (*L*^*2* ∗ (*k* − *1*) ^ *m*)) / *2* ^ (*p* − *1*)
shows ((*m* − *l*) / *k*)^*l* / (*6* ∗ *L*^*2*) < *s*
⟨*proof*⟩

lemma *identities*: *k* = *root 4 m l* = *root 8 m*
⟨*proof*⟩

lemma *identities2*: *root 4 m* = *m powr* (*1*/*4*) *root 8 m* = *m powr* (*1*/*8*)
⟨*proof*⟩

lemma *appendix-A-1*: **assumes** *x* ≥ *M0′* **shows** *x powr* (*2*/*3*) ≤ *x powr* (*3*/*4*) −

1
⟨*proof*⟩

lemma *appendix-A-2*: $(p - 1)\hat{}l < m\ powr\ ((1 / 8 + eps) * l)$
⟨*proof*⟩

lemma *appendix-A-3*: $6 * real\ l\hat{}16 * fact\ l < m\ powr\ (1 / 8 * l)$
⟨*proof*⟩

lemma *appendix-A-4*: $12 * L\hat{}2 \leq m\ powr\ (m\ powr\ (1 / 8) * 0.51)$
⟨*proof*⟩

lemma *approximation4*: **fixes** $s :: nat$
assumes $s > ((m - l) / k)\hat{}l / (6 * L\hat{}2)$
shows $s > 2 * k\ powr\ (4 / 7 * sqrt\ k)$
⟨*proof*⟩

end

end
theory *Clique-Large-Monotone-Circuits*
imports
Sunflowers.Erdos-Rado-Sunflower
Preliminaries
Assumptions-and-Approximations
Monotone-Formula
begin

disable list-syntax

no-syntax *-list* :: *args* ⇒ *'a list* ([(-)])
no-syntax *--listcompr* :: *args* ⇒ *'a list* ([(-)])

hide-const (**open**) *Sigma-Algebra.measure*

4.2 Plain Graphs

definition *binprod* :: *'a set* ⇒ *'a set* ⇒ *'a set set* (**infixl** · *60*) **where**
$X \cdot Y = \{\{x,y\} \mid x\ y.\ x \in X \wedge y \in Y \wedge x \neq y\}$

abbreviation *sameprod* :: *'a set* ⇒ *'a set set* ((-)^**2**) **where**
$X\hat{}\mathbf{2} \equiv X \cdot X$

lemma *sameprod-altdef*: $X\hat{}\mathbf{2} = \{Y.\ Y \subseteq X \wedge card\ Y = 2\}$
⟨*proof*⟩

definition *numbers* :: *nat* ⇒ *nat set* ([(-)]) **where**
$[n] \equiv \{..<n\}$

lemma *card-sameprod*: *finite* $X \Longrightarrow$ *card* $(X \hat{} \mathbf{2}) =$ *card* X *choose 2*
⟨*proof*⟩

lemma *sameprod-mono*: $X \subseteq Y \Longrightarrow X \hat{} \mathbf{2} \subseteq Y \hat{} \mathbf{2}$
⟨*proof*⟩

lemma *sameprod-finite*: *finite* $X \Longrightarrow$ *finite* $(X \hat{} \mathbf{2})$
⟨*proof*⟩

lemma *numbers2-mono*: $x \leq y \Longrightarrow [x] \hat{} \mathbf{2} \subseteq [y] \hat{} \mathbf{2}$
⟨*proof*⟩

lemma *card-numbers*[*simp*]: *card* $[n] = n$
⟨*proof*⟩

lemma *card-numbers2*[*simp*]: *card* $([n] \hat{} \mathbf{2}) = n$ *choose 2*
⟨*proof*⟩

type-synonym *vertex* = *nat*
type-synonym *graph* = *vertex set set*

definition *Graphs* :: *vertex set* ⇒ *graph set* **where**
Graphs $V = \{\ G.\ G \subseteq V \hat{} \mathbf{2}\ \}$

definition *Clique* :: *vertex set* ⇒ *nat* ⇒ *graph set* **where**
Clique $V\ k = \{\ G.\ G \in$ *Graphs* $V \wedge (\exists\ C \subseteq V.\ C \hat{} \mathbf{2} \subseteq G \wedge$ *card* $C = k)\ \}$

context *first-assumptions*
begin

abbreviation $\mathcal{G}$ **where** $\mathcal{G} \equiv$ *Graphs* $[m]$

lemmas $\mathcal{G}$*-def* = *Graphs-def*[*of* $[m]$]

lemma *empty-*$\mathcal{G}$[*simp*]: $\{\} \in \mathcal{G}$ ⟨*proof*⟩

definition v :: *graph* ⇒ *vertex set* **where**
$v\ G = \{\ x\ .\ \exists\ y.\ \{x,y\} \in G\}$

lemma *v-union*: $v\ (G \cup H) = v\ G \cup v\ H$
⟨*proof*⟩

definition $\mathcal{K}$ :: *graph set* **where**
$\mathcal{K} = \{\ K\ .\ K \in \mathcal{G} \wedge$ *card* $(v\ K) = k \wedge K = (v\ K) \hat{} \mathbf{2}\ \}$

lemma *v-*$\mathcal{G}$: $G \in \mathcal{G} \Longrightarrow v\ G \subseteq [m]$
⟨*proof*⟩

lemma *v-mono*: $G \subseteq H \Longrightarrow v\ G \subseteq v\ H$ ⟨*proof*⟩

lemma *v-sameprod*[*simp*]: **assumes** *card* $X \geq 2$
shows $v\ (X\hat{}\mathbf{2}) = X$
⟨*proof*⟩

lemma *v-mem-sub*: **assumes** *card* $e = 2$ $e \in G$ **shows** $e \subseteq v\ G$
⟨*proof*⟩

lemma *v-$\mathcal{G}$-2*: **assumes** $G \in \mathcal{G}$ **shows** $G \subseteq (v\ G)\hat{}\mathbf{2}$
⟨*proof*⟩

lemma *v-numbers2*[*simp*]: $x \geq 2 \Longrightarrow v\ ([x]\hat{}\mathbf{2}) = [x]$
⟨*proof*⟩

lemma *sameprod-$\mathcal{G}$*: **assumes** $X \subseteq [m]$ *card* $X \geq 2$
shows $X\hat{}\mathbf{2} \in \mathcal{G}$
⟨*proof*⟩

lemma *finite-numbers*[*simp*,*intro*]: *finite* $[n]$
⟨*proof*⟩

lemma *finite-numbers2*[*simp*,*intro*]: *finite* $([n]\hat{}\mathbf{2})$
⟨*proof*⟩

lemma *finite-members-$\mathcal{G}$*: $G \in \mathcal{G} \Longrightarrow$ *finite* G
⟨*proof*⟩

lemma *finite-$\mathcal{G}$*[*simp*,*intro*]: *finite* $\mathcal{G}$
⟨*proof*⟩

lemma *finite-vG*: **assumes** $G \in \mathcal{G}$
shows *finite* $(v\ G)$
⟨*proof*⟩

lemma *v-empty*[*simp*]: $v\ \{\} = \{\}$ ⟨*proof*⟩

lemma *v-card2*: **assumes** $G \in \mathcal{G}$ $G \neq \{\}$
shows $2 \leq$ *card* $(v\ G)$
⟨*proof*⟩

lemma *$\mathcal{K}$-altdef*: $\mathcal{K} = \{V\hat{}\mathbf{2} \mid V.\ V \subseteq [m] \wedge$ *card* $V = k\}$
(**is** - = *?R*)
⟨*proof*⟩

lemma *$\mathcal{K}$-$\mathcal{G}$*: $\mathcal{K} \subseteq \mathcal{G}$
⟨*proof*⟩

definition *CLIQUE* :: *graph set* **where**
CLIQUE = { *G*. *G* ∈ $\mathcal{G}$ ∧ (∃ *K* ∈ $\mathcal{K}$. *K* ⊆ *G*) }

lemma *empty-CLIQUE*[*simp*]: {} ∉ *CLIQUE* ⟨*proof*⟩

4.3 Test Graphs

Positive test graphs are precisely the cliques of size *k*.

abbreviation *POS* ≡ $\mathcal{K}$

lemma *POS-*$\mathcal{G}$: *POS* ⊆ $\mathcal{G}$ ⟨*proof*⟩

Negative tests are coloring-functions of vertices that encode graphs which have cliques of size at most *k* − *1*.

type-synonym *colorf* = *vertex* ⇒ *nat*

definition $\mathcal{F}$ :: *colorf set* **where**
$\mathcal{F}$ = [*m*] →$_E$ [*k* − *1*]

lemma *finite-*$\mathcal{F}$: *finite* $\mathcal{F}$
⟨*proof*⟩

definition *C* :: *colorf* ⇒ *graph* **where**
C f = { {*x*, *y*} | *x y* . {*x*,*y*} ∈ [*m*]^**2** ∧ *f x* ≠ *f y*}

definition *NEG* :: *graph set* **where**
NEG = *C* ‘ $\mathcal{F}$

Lemma 1 **lemma** *CLIQUE-NEG*: *CLIQUE* ∩ *NEG* = {}
⟨*proof*⟩

lemma *NEG-*$\mathcal{G}$: *NEG* ⊆ $\mathcal{G}$
⟨*proof*⟩

lemma *finite-POS-NEG*: *finite* (*POS* ∪ *NEG*)
⟨*proof*⟩

lemma *POS-sub-CLIQUE*: *POS* ⊆ *CLIQUE*
⟨*proof*⟩

lemma *POS-CLIQUE*: *POS* ⊂ *CLIQUE*
⟨*proof*⟩

lemma *card-POS*: *card POS* = *m choose k*
⟨*proof*⟩

4.4 Basic operations on sets of graphs

definition *odot* :: *graph set* ⇒ *graph set* ⇒ *graph set* (**infixl** ⊙ *65*) **where**
$X \odot Y = \{ D \cup E \mid D\ E.\ D \in X \wedge E \in Y \}$

lemma *union-𝒢*[*intro*]: $G \in \mathcal{G} \Longrightarrow H \in \mathcal{G} \Longrightarrow G \cup H \in \mathcal{G}$
⟨*proof*⟩

lemma *odot-𝒢*: $X \subseteq \mathcal{G} \Longrightarrow Y \subseteq \mathcal{G} \Longrightarrow X \odot Y \subseteq \mathcal{G}$
⟨*proof*⟩

4.5 Acceptability

Definition 2

definition *accepts* :: *graph set* ⇒ *graph* ⇒ *bool* (**infixl** ⊩ *55*) **where**
$(X \Vdash G) = (\exists\ D \in X.\ D \subseteq G)$

lemma *acceptsI*[*intro*]: $D \subseteq G \Longrightarrow D \in X \Longrightarrow X \Vdash G$
⟨*proof*⟩

definition *ACC* :: *graph set* ⇒ *graph set* **where**
$ACC\ X = \{ G.\ G \in \mathcal{G} \wedge X \Vdash G \}$

definition *ACC-cf* :: *graph set* ⇒ *colorf set* **where**
$ACC\text{-}cf\ X = \{ F.\ F \in \mathcal{F} \wedge X \Vdash C\ F \}$

lemma *ACC-cf-ℱ*: $ACC\text{-}cf\ X \subseteq \mathcal{F}$
⟨*proof*⟩

lemma *finite-ACC*[*intro*,*simp*]: *finite* $(ACC\text{-}cf\ X)$
⟨*proof*⟩

lemma *ACC-I*[*intro*]: $G \in \mathcal{G} \Longrightarrow X \Vdash G \Longrightarrow G \in ACC\ X$
⟨*proof*⟩

lemma *ACC-cf-I*[*intro*]: $F \in \mathcal{F} \Longrightarrow X \Vdash C\ F \Longrightarrow F \in ACC\text{-}cf\ X$
⟨*proof*⟩

lemma *ACC-cf-mono*: $X \subseteq Y \Longrightarrow ACC\text{-}cf\ X \subseteq ACC\text{-}cf\ Y$
⟨*proof*⟩

Lemma 3

lemma *ACC-cf-empty*: $ACC\text{-}cf\ \{\} = \{\}$
⟨*proof*⟩

lemma *ACC-empty*[*simp*]: $ACC\ \{\} = \{\}$
⟨*proof*⟩

lemma *ACC-cf-union*: $ACC\text{-}cf\ (X \cup Y) = ACC\text{-}cf\ X \cup ACC\text{-}cf\ Y$
⟨*proof*⟩

lemma *ACC-union*: $ACC\ (X \cup Y) = ACC\ X \cup ACC\ Y$
⟨*proof*⟩

lemma *ACC-odot*: $ACC\ (X \odot Y) = ACC\ X \cap ACC\ Y$
⟨*proof*⟩

lemma *ACC-cf-odot*: $ACC\text{-}cf\ (X \odot Y) = ACC\text{-}cf\ X \cap ACC\text{-}cf\ Y$
⟨*proof*⟩

4.6 Approximations and deviations

definition $\mathcal{G}l$ :: *graph set* **where**
$\mathcal{G}l = \{\ G.\ G \in \mathcal{G} \wedge card\ (v\ G) \leq l\ \}$

definition *v-gs* :: *graph set* ⇒ *vertex set set* **where**
$v\text{-}gs\ X = v\ `\ X$

lemma *v-gs-empty*[*simp*]: $v\text{-}gs\ \{\} = \{\}$
⟨*proof*⟩

lemma *v-gs-union*: $v\text{-}gs\ (X \cup Y) = v\text{-}gs\ X \cup v\text{-}gs\ Y$
⟨*proof*⟩

lemma *v-gs-mono*: $X \subseteq Y \Longrightarrow v\text{-}gs\ X \subseteq v\text{-}gs\ Y$
⟨*proof*⟩

lemma *finite-v-gs*: **assumes** $X \subseteq \mathcal{G}$
shows *finite* $(v\text{-}gs\ X)$
⟨*proof*⟩

lemma *finite-v-gs-Gl*: **assumes** $X \subseteq \mathcal{G}l$
shows *finite* $(v\text{-}gs\ X)$
⟨*proof*⟩

definition $\mathcal{P}L\mathcal{G}l$ :: *graph set set* **where**
$\mathcal{P}L\mathcal{G}l = \{\ X\ .\ X \subseteq \mathcal{G}l \wedge card\ (v\text{-}gs\ X) \leq L\}$

definition *odotl* :: *graph set* ⇒ *graph set* ⇒ *graph set* (**infixl** $\odot l$ *65*) **where**
$X \odot l\ Y = \{D \cup E \mid D\ E.\ D \in X \wedge E \in Y \wedge D \cup E \in \mathcal{G}l\}$

lemma *joinl-join*: $X \odot l\ Y \subseteq X \odot Y$
⟨*proof*⟩

lemma *card-v-gs-join*: **assumes** *X*: $X \subseteq \mathcal{G}$ **and** *Y*: $Y \subseteq \mathcal{G}$

and Z: $Z \subseteq X \odot Y$
shows *card* (*v-gs* Z) $\leq$ *card* (*v-gs* X) $*$ *card* (*v-gs* Y)
⟨*proof*⟩

Definition 6 – elementary plucking step

definition *plucking-step* :: *graph set* ⇒ *graph set* **where**
plucking-step X = (*let* vXp = *v-gs* X;
S = (*SOME* S. $S \subseteq vXp \wedge$ *sunflower* $S \wedge$ *card* $S = p$);
$U = \{E \in X.\ v\ E \in S\}$;
$Vs = \bigcap S$;
Gs = Vs^**2**
in $X - U \cup \{Gs\}$)
end

context *second-assumptions*
begin

Lemma 9 – for elementary plucking step

lemma *v-sameprod-subset*: v (Vs^**2**) $\subseteq$ Vs ⟨*proof*⟩

lemma *plucking-step*: **assumes** X: $X \subseteq \mathcal{G}l$
and L: *card* (*v-gs* X) > L
and Y: Y = *plucking-step* X
shows *card* (*v-gs* Y) $\leq$ *card* (*v-gs* X) − p + *1*
$Y \subseteq \mathcal{G}l$
POS ∩ *ACC* $X \subseteq$ *ACC* Y
2 ^ p $*$ *card* (*ACC-cf* Y − *ACC-cf* X) $\leq$ (k − *1*) ^ m
$Y \neq \{\}$
⟨*proof*⟩

Definition 6

function *PLU-main* :: *graph set* ⇒ *graph set* × *nat* **where**
PLU-main X = (*if* $X \subseteq \mathcal{G}l \wedge L <$ *card* (*v-gs* X) *then*
map-prod id Suc (*PLU-main* (*plucking-step* X)) *else*
(X, *0*))
⟨*proof*⟩

termination
⟨*proof*⟩

declare *PLU-main.simps*[*simp del*]

definition *PLU* :: *graph set* ⇒ *graph set* **where**
PLU X = *fst* (*PLU-main* X)

Lemma 7

lemma *PLU-main-n*: **assumes** $X \subseteq \mathcal{G}l$ **and** *PLU-main* X = (Z, n)
shows $n * (p - 1) \leq$ *card* (*v-gs* X)

⟨*proof*⟩

Definition 8

definition *sqcup* :: *graph set* ⇒ *graph set* ⇒ *graph set* (**infixl** ⊔ *65*) **where**
X ⊔ *Y* = *PLU* (*X* ∪ *Y*)

definition *sqcap* :: *graph set* ⇒ *graph set* ⇒ *graph set* (**infixl** ⊓ *65*) **where**
X ⊓ *Y* = *PLU* (*X* ⊙*l* *Y*)

definition *deviate-pos-cup* :: *graph set* ⇒ *graph set* ⇒ *graph set* (∂⊔*Pos*) **where**
∂⊔*Pos X Y* = *POS* ∩ *ACC* (*X* ∪ *Y*) − *ACC* (*X* ⊔ *Y*)

definition *deviate-pos-cap* :: *graph set* ⇒ *graph set* ⇒ *graph set* (∂⊓*Pos*) **where**
∂⊓*Pos X Y* = *POS* ∩ *ACC* (*X* ⊙ *Y*) − *ACC* (*X* ⊓ *Y*)

definition *deviate-neg-cup* :: *graph set* ⇒ *graph set* ⇒ *colorf set* (∂⊔*Neg*) **where**
∂⊔*Neg X Y* = *ACC-cf* (*X* ⊔ *Y*) − *ACC-cf* (*X* ∪ *Y*)

definition *deviate-neg-cap* :: *graph set* ⇒ *graph set* ⇒ *colorf set* (∂⊓*Neg*) **where**
∂⊓*Neg X Y* = *ACC-cf* (*X* ⊓ *Y*) − *ACC-cf* (*X* ⊙ *Y*)

Lemma 9 – without applying Lemma 7

lemma *PLU-main*: **assumes** *X* ⊆ *Gl*
and *PLU-main X* = (*Z*, *n*)
shows *Z* ∈ *PLGl*
∧ (*Z* = {} ⟷ *X* = {})
∧ *POS* ∩ *ACC X* ⊆ *ACC Z*
∧ *2* ^ *p* ∗ *card* (*ACC-cf Z* − *ACC-cf X*) ≤ (*k* − *1*) ^ *m* ∗ *n*
⟨*proof*⟩

Lemma 9

lemma assumes *X*: *X* ∈ *PLGl* **and** *Y*: *Y* ∈ *PLGl*
shows *PLU-union*: *PLU* (*X* ∪ *Y*) ∈ *PLGl* **and**
sqcup: *X* ⊔ *Y* ∈ *PLGl* **and**
sqcup-sub: *POS* ∩ *ACC* (*X* ∪ *Y*) ⊆ *ACC* (*X* ⊔ *Y*) **and**
deviate-pos-cup: ∂⊔*Pos X Y* = {} **and**
deviate-neg-cup: *card* (∂⊔*Neg X Y*) < (*k* − *1*)^*m* ∗ *L* / *2*^(*p* − *1*)
⟨*proof*⟩

Lemma 10

lemma assumes *X*: *X* ∈ *PLGl* **and** *Y*: *Y* ∈ *PLGl*
shows *PLU-joinl*: *PLU* (*X* ⊙*l* *Y*) ∈ *PLGl* **and**
sqcap: *X* ⊓ *Y* ∈ *PLGl* **and**
deviate-neg-cap: *card* (∂⊓*Neg X Y*) < (*k* − *1*)^*m* ∗ *L*^*2* / *2*^(*p* − *1*) **and**
deviate-pos-cap: *card* (∂⊓*Pos X Y*) ≤ ((*m* − *l* − *1*) *choose* (*k* − *l* − *1*)) ∗ *L*^*2*
⟨*proof*⟩
end

4.7 Formalism

Fix a variable set of cardinality m over 2.

locale *forth-assumptions* = *third-assumptions* +
  fixes $\mathcal{V}$ :: *'a set* **and** π :: *'a* $\Rightarrow$ *vertex set*
  assumes *cV*: *card* $\mathcal{V}$ = (*m choose 2*)
  and *bij-betw-π*: *bij-betw* π $\mathcal{V}$ ([*m*]^**2**)
begin

definition *n* **where** *n* = (*m choose 2*)

the formulas over the fixed variable set

definition $\mathcal{A}$ :: *'a mformula set* **where**
  $\mathcal{A}$ = { φ. *vars* $\varphi \subseteq \mathcal{V}$}

lemma $\mathcal{A}$*-simps*[*simp*]:
  FALSE $\in \mathcal{A}$
  (*Var x* $\in \mathcal{A}$) = ($x \in \mathcal{V}$)
  (*Conj* φ $\psi \in \mathcal{A}$) = ($\varphi \in \mathcal{A} \wedge \psi \in \mathcal{A}$)
  (*Disj* φ $\psi \in \mathcal{A}$) = ($\varphi \in \mathcal{A} \wedge \psi \in \mathcal{A}$)
  ⟨*proof*⟩

lemma *inj-on-π*: *inj-on* π $\mathcal{V}$
  ⟨*proof*⟩

lemma π*m2*[*simp*,*intro*]: $x \in \mathcal{V} \Longrightarrow \pi\ x \in$ [*m*]^**2**
  ⟨*proof*⟩

lemma *card-v-π*[*simp*,*intro*]: **assumes** $x \in \mathcal{V}$
  shows *card* (*v* {π *x*}) = *2*
⟨*proof*⟩

lemma π*-singleton*[*simp*,*intro*]: **assumes** $x \in \mathcal{V}$
  shows {π *x*} $\in \mathcal{G}$
    {{π *x*}} $\in \mathcal{PLG}l$
  ⟨*proof*⟩

lemma *empty-*$\mathcal{PLG}l$[*simp*,*intro*]: {} $\in \mathcal{PLG}l$
  ⟨*proof*⟩

fun *SET* :: *'a mformula* $\Rightarrow$ *graph set* **where**
  SET FALSE = {}
| *SET* (*Var x*) = {{π *x*}}
| *SET* (*Disj* φ ψ) = *SET* φ $\cup$ *SET* ψ
| *SET* (*Conj* φ ψ) = *SET* φ $\odot$ *SET* ψ

lemma *ACC-cf-SET*[*simp*]:
  ACC-cf (*SET* (*Var x*)) = {$f \in \mathcal{F}$. $\pi\ x \in C\ f$}
  ACC-cf (*SET FALSE*) = {}

$ACC\text{-}cf\ (SET\ (Disj\ \varphi\ \psi)) = ACC\text{-}cf\ (SET\ \varphi) \cup ACC\text{-}cf\ (SET\ \psi)$
$ACC\text{-}cf\ (SET\ (Conj\ \varphi\ \psi)) = ACC\text{-}cf\ (SET\ \varphi) \cap ACC\text{-}cf\ (SET\ \psi)$
⟨*proof*⟩

lemma *ACC-SET*[*simp*]:
$ACC\ (SET\ (Var\ x)) = \{G \in \mathcal{G}.\ \pi\ x \in G\}$
$ACC\ (SET\ FALSE) = \{\}$
$ACC\ (SET\ (Disj\ \varphi\ \psi)) = ACC\ (SET\ \varphi) \cup ACC\ (SET\ \psi)$
$ACC\ (SET\ (Conj\ \varphi\ \psi)) = ACC\ (SET\ \varphi) \cap ACC\ (SET\ \psi)$
⟨*proof*⟩

lemma *SET-𝒢*: $\varphi \in$ *tf-mformula* $\Longrightarrow \varphi \in \mathcal{A} \Longrightarrow SET\ \varphi \subseteq \mathcal{G}$
⟨*proof*⟩

fun $APR :: 'a\ mformula \Rightarrow graph\ set$ **where**
$APR\ FALSE = \{\}$
| $APR\ (Var\ x) = \{\{\pi\ x\}\}$
| $APR\ (Disj\ \varphi\ \psi) = APR\ \varphi \sqcup APR\ \psi$
| $APR\ (Conj\ \varphi\ \psi) = APR\ \varphi \sqcap APR\ \psi$

lemma *APR*: $\varphi \in$ *tf-mformula* $\Longrightarrow \varphi \in \mathcal{A} \Longrightarrow APR\ \varphi \in \mathcal{PLG}l$
⟨*proof*⟩

definition $ACC\text{-}cf\text{-}mf :: 'a\ mformula \Rightarrow colorf\ set$ **where**
$ACC\text{-}cf\text{-}mf\ \varphi = ACC\text{-}cf\ (SET\ \varphi)$

definition $ACC\text{-}mf :: 'a\ mformula \Rightarrow graph\ set$ **where**
$ACC\text{-}mf\ \varphi = ACC\ (SET\ \varphi)$

definition $deviate\text{-}pos :: 'a\ mformula \Rightarrow graph\ set\ (\partial Pos)$ **where**
$\partial Pos\ \varphi = POS \cap ACC\text{-}mf\ \varphi - ACC\ (APR\ \varphi)$

definition $deviate\text{-}neg :: 'a\ mformula \Rightarrow colorf\ set\ (\partial Neg)$ **where**
$\partial Neg\ \varphi = ACC\text{-}cf\ (APR\ \varphi) - ACC\text{-}cf\text{-}mf\ \varphi$

Lemma 11.1

lemma *deviate-subset-Disj*:
$\partial Pos\ (Disj\ \varphi\ \psi) \subseteq \partial\sqcup Pos\ (APR\ \varphi)\ (APR\ \psi) \cup \partial Pos\ \varphi \cup \partial Pos\ \psi$
$\partial Neg\ (Disj\ \varphi\ \psi) \subseteq \partial\sqcup Neg\ (APR\ \varphi)\ (APR\ \psi) \cup \partial Neg\ \varphi \cup \partial Neg\ \psi$
⟨*proof*⟩

Lemma 11.2

lemma *deviate-subset-Conj*:
$\partial Pos\ (Conj\ \varphi\ \psi) \subseteq \partial\sqcap Pos\ (APR\ \varphi)\ (APR\ \psi) \cup \partial Pos\ \varphi \cup \partial Pos\ \psi$
$\partial Neg\ (Conj\ \varphi\ \psi) \subseteq \partial\sqcap Neg\ (APR\ \varphi)\ (APR\ \psi) \cup \partial Neg\ \varphi \cup \partial Neg\ \psi$
⟨*proof*⟩

lemmas *deviate-subset* = *deviate-subset-Disj deviate-subset-Conj*

lemma *deviate-finite*:
finite (*∂Pos φ*)
finite (*∂Neg φ*)
finite (*∂⊔Pos A B*)
finite (*∂⊔Neg A B*)
finite (*∂⊓Pos A B*)
finite (*∂⊓Neg A B*)
⟨*proof*⟩

Lemma 12

lemma *no-deviation*[*simp*]:
∂Pos FALSE = {}
∂Neg FALSE = {}
∂Pos (*Var x*) = {}
∂Neg (*Var x*) = {}
⟨*proof*⟩

Lemma 12.1-2

fun *approx-pos* **where**
approx-pos (*Conj phi psi*) = *∂⊓Pos* (*APR phi*) (*APR psi*)
| *approx-pos* - = {}

fun *approx-neg* **where**
approx-neg (*Conj phi psi*) = *∂⊓Neg* (*APR phi*) (*APR psi*)
| *approx-neg* (*Disj phi psi*) = *∂⊔Neg* (*APR phi*) (*APR psi*)
| *approx-neg* - = {}

lemma *finite-approx-pos*: *finite* (*approx-pos φ*)
⟨*proof*⟩

lemma *finite-approx-neg*: *finite* (*approx-neg φ*)
⟨*proof*⟩

lemma *card-deviate-Pos*: **assumes** *phi*: $\varphi \in$ *tf-mformula* $\varphi \in \mathcal{A}$
shows *card* (*∂Pos φ*) ≤ *cs φ* ∗ L^2 ∗ ((*m* − *l* − *1*) *choose* (*k* − *l* − *1*))
⟨*proof*⟩

lemma *card-deviate-Neg*: **assumes** *phi*: $\varphi \in$ *tf-mformula* $\varphi \in \mathcal{A}$
shows *card* (*∂Neg φ*) ≤ *cs φ* ∗ L^2 ∗ (*k* − *1*)^*m* / *2*^(*p* − *1*)
⟨*proof*⟩

Lemma 12.3

lemma *ACC-cf-non-empty-approx*: **assumes** *phi*: $\varphi \in$ *tf-mformula* $\varphi \in \mathcal{A}$
and *ne*: *APR φ* ≠ {}
shows *card* (*ACC-cf* (*APR φ*)) > (*k* − *1*)^*m* / *3*
⟨*proof*⟩

Theorem 13

lemma *theorem-13*: **assumes** *phi*: $\varphi \in$ *tf-mformula* $\varphi \in \mathcal{A}$

and *sub*: $POS \subseteq ACC\text{-}mf\ \varphi\ ACC\text{-}cf\text{-}mf\ \varphi = \{\}$
shows $cs\ \varphi > k\ powr\ (4\ /\ 7 * sqrt\ k)$
⟨*proof*⟩

Definition 14

definition *eval-g* :: *'a VAS* ⇒ *graph* ⇒ *bool* **where**
$eval\text{-}g\ \vartheta\ G = (\forall\ v \in \mathcal{V}.\ (\pi\ v \in G \longrightarrow \vartheta\ v))$

definition *eval-gs* :: *'a VAS* ⇒ *graph set* ⇒ *bool* **where**
$eval\text{-}gs\ \vartheta\ X = (\exists\ G \in X.\ eval\text{-}g\ \vartheta\ G)$

lemmas *eval-simps* = *eval-g-def eval-gs-def eval.simps*

lemma *eval-gs-union*:
$eval\text{-}gs\ \vartheta\ (X \cup Y) = (eval\text{-}gs\ \vartheta\ X \vee eval\text{-}gs\ \vartheta\ Y)$
⟨*proof*⟩

lemma *eval-gs-odot*: **assumes** $X \subseteq \mathcal{G}\ Y \subseteq \mathcal{G}$
shows $eval\text{-}gs\ \vartheta\ (X \odot Y) = (eval\text{-}gs\ \vartheta\ X \wedge eval\text{-}gs\ \vartheta\ Y)$
⟨*proof*⟩

Lemma 15

lemma *eval-set*: **assumes** *phi*: $\varphi \in tf\text{-}mformula\ \varphi \in \mathcal{A}$
shows $eval\ \vartheta\ \varphi = eval\text{-}gs\ \vartheta\ (SET\ \varphi)$
⟨*proof*⟩

definition ϑ_g :: *graph* ⇒ *'a VAS* **where**
$\vartheta_g\ G\ x = (x \in \mathcal{V} \wedge \pi\ x \in G)$

From here on we deviate from Gordeev's paper as we do not use positive bases, but a more direct approach.

lemma *eval-ACC*: **assumes** *phi*: $\varphi \in tf\text{-}mformula\ \ \varphi \in \mathcal{A}$
and *G*: $G \in \mathcal{G}$
shows $eval\ (\vartheta_g\ G)\ \varphi = (G \in ACC\text{-}mf\ \varphi)$
⟨*proof*⟩

lemma *CLIQUE-solution-imp-POS-sub-ACC*: **assumes** *solution*: $\forall\ G \in \mathcal{G}.\ G \in CLIQUE \longleftrightarrow eval\ (\vartheta_g\ G)\ \varphi$
and *tf*: $\varphi \in tf\text{-}mformula$
and *phi*: $\varphi \in \mathcal{A}$
shows $POS \subseteq ACC\text{-}mf\ \varphi$
⟨*proof*⟩

lemma *CLIQUE-solution-imp-ACC-cf-empty*: **assumes** *solution*: $\forall\ G \in \mathcal{G}.\ G \in CLIQUE \longleftrightarrow eval\ (\vartheta_g\ G)\ \varphi$
and *tf*: $\varphi \in tf\text{-}mformula$
and *phi*: $\varphi \in \mathcal{A}$
shows $ACC\text{-}cf\text{-}mf\ \varphi = \{\}$

⟨*proof*⟩

4.8 Conclusion

Theorem 22

We first consider monotone formulas without TRUE.

theorem *Clique-not-solvable-by-small-tf-mformula*: **assumes** *solution*: $\forall\ G \in \mathcal{G}.$ $G \in CLIQUE \longleftrightarrow eval\ (\vartheta_g\ G)\ \varphi$
and *tf*: $\varphi \in$ *tf-mformula*
and *phi*: $\varphi \in \mathcal{A}$
shows *cs* $\varphi > k$ *powr* $(4\ /\ 7 * sqrt\ k)$
⟨*proof*⟩

Next we consider general monotone formulas.

theorem *Clique-not-solvable-by-poly-mono*: **assumes** *solution*: $\forall\ G \in \mathcal{G}.\ G \in CLIQUE \longleftrightarrow eval\ (\vartheta_g\ G)\ \varphi$
and *phi*: $\varphi \in \mathcal{A}$
shows *cs* $\varphi > k$ *powr* $(4\ /\ 7 * sqrt\ k)$
⟨*proof*⟩

We next expand all abbreviations and definitions of the locale, but stay within the locale

theorem *Clique-not-solvable-by-small-monotone-circuit-in-locale*: **assumes** *phi-solves-clique*:

$\forall\ G \in Graphs\ [k\hat{}4].\ G \in Clique\ [k\hat{}4]\ k \longleftrightarrow eval\ (\lambda\ x.\ \pi\ x \in G)\ \varphi$
and *vars*: *vars* $\varphi \subseteq \mathcal{V}$
shows *cs* $\varphi > k$ *powr* $(4\ /\ 7 * sqrt\ k)$
⟨*proof*⟩
end

Let us now move the theorem outside the locale

definition *Large-Number* **where** *Large-Number* = *Max* {*64*, *L0''*^*2*, *L0*^*2*, *L0'*^*2*, *M0*, *M0'*}

theorem *Clique-not-solvable-by-small-monotone-circuit-squared*:
fixes φ :: *'a mformula*
assumes *k*: $\exists\ l.\ k = l\hat{}2$
and *LARGE*: $k \geq$ *Large-Number*
and π: *bij-betw* π *V* $[k\hat{}4]\hat{}\mathbf{2}$
and *solution*: $\forall\ G \in Graphs\ [k\ \hat{}\ 4].\ (G \in Clique\ [k\ \hat{}\ 4]\ k) = eval\ (\lambda\ x.\ \pi\ x \in G)\ \varphi$
and *vars*: *vars* $\varphi \subseteq$ *V*
shows *cs* $\varphi > k$ *powr* $(4\ /\ 7 * sqrt\ k)$
⟨*proof*⟩

A variant where we get rid of the $k = l^2$-assumption by just taking squares everywhere.

theorem *Clique-not-solvable-by-small-monotone-circuit*:
fixes φ :: *'a mformula*
assumes *LARGE*: $k \geq$ *Large-Number*
and π: *bij-betw* π *V* [*k*^*8*]^**2**
and *solution*: $\forall$ *G*∈*Graphs* [*k* ^ *8*]. (*G* ∈ *Clique* [*k* ^ *8*] (*k*^*2*)) = *eval* (λ *x*. π *x* ∈ *G*) φ
and *vars*: *vars* $\varphi \subseteq$ *V*
shows *cs* $\varphi >$ *k powr* (*8* / *7* ∗ *k*)
⟨*proof*⟩

definition *large-number* **where** *large-number* = *Large-Number*^*8*

Finally a variant, where the size is formulated depending on n, the number of vertices.

theorem *Clique-with-n-nodes-not-solvable-by-small-monotone-circuit*:
fixes φ :: *'a mformula*
assumes *large*: $n \geq$ *large-number*
and *kn*: ∃ *k*. *n* = *k*^*8*
and π: *bij-betw* π *V* [*n*]^**2**
and *s*: *s* = *root 4 n*
and *solution*: $\forall$ *G*∈*Graphs* [*n*]. (*G* ∈ *Clique* [*n*] *s*) = *eval* (λ *x*. π *x* ∈ *G*) φ
and *vars*: *vars* $\varphi \subseteq$ *V*
shows *cs* $\varphi >$ (*root 7 n*) *powr* (*root 8 n*)
⟨*proof*⟩

end

References

[1] N. Alon and R. B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] R. B. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 757–804. Elsevier and MIT Press, 1990.

[3] P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960.

[4] L. Gordeev. On P versus NP. Avaible at http://arxiv.org/abs/2005.00809v3.

[5] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.