

Clique is not solvable by monotone circuits of polynomial size*

René Thiemann
University of Innsbruck

July 1, 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 $\sqrt[3]{n}^{\sqrt[3]{n}}$ for the fixed choice of $s = \sqrt[4]{n}$), following a proof by Gordeev.

Contents

1	Introduction	2
2	Preliminaries	2
3	Monotone Formulas	3
3.1	Definition	4
3.2	Conversion of mformulas to true-free mformulas	4
4	Simplified Version of Gordeev’s Proof for Monotone Circuits	5
4.1	Setup of Global Assumptions and Proofs of Approximations .	5
4.2	Plain Graphs	8
4.3	Test Graphs	11
4.4	Basic operations on sets of graphs	12
4.5	Acceptability	12
4.6	Approximations and deviations	13
4.7	Formalism	16
4.8	Conclusion	20

*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We arrived at this formalization by trying to verify an unpublished draft of Gordeev [4], which tries to show that Clique cannot be solved by any polynomial-sized circuit, including non-monotone ones, where the concrete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 $\exists B. A \subseteq B \wedge B \subseteq C \wedge \text{card } B = n$

<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 $\text{fact } k \leq \text{fact } (k - n) * (\text{real } k \hat{=} n)$
 $\langle \text{proof} \rangle$

lemma *fact-approx-upper-add*: **assumes** $al: a \leq \text{Suc } l$ **shows** $\text{fact } l * \text{real } a \hat{=} n$
 $\leq \text{fact } (l + n)$
 $\langle \text{proof} \rangle$

lemma *fact-approx-upper-minus*: **assumes** $n \leq k$ **and** $n + a \leq \text{Suc } k$
shows $\text{fact } (k - n) * \text{real } a \hat{=} n \leq \text{fact } k$
 $\langle \text{proof} \rangle$

lemma *choose-mono*: $n \leq m \implies n \text{ choose } k \leq m \text{ choose } k$
 $\langle \text{proof} \rangle$

lemma *div-mult-le*: $(a \text{ div } b) * c \leq (a * c) \text{ div } (b :: \text{nat})$
 $\langle \text{proof} \rangle$

lemma *div-mult-pow-le*: $(a \text{ div } b) \hat{=} n \leq a \hat{=} n \text{ div } (b :: \text{nat}) \hat{=} n$
 $\langle \text{proof} \rangle$

lemma *choose-inj-right*:
assumes $\text{id}: (n \text{ choose } l) = (k \text{ choose } l)$
and $n0: n \text{ choose } l \neq 0$
and $l0: l \neq 0$
shows $n = k$
 $\langle \text{proof} \rangle$

lemma *card-funcsetE*: $\text{finite } A \implies \text{card } (A \rightarrow_E B) = \text{card } B \hat{=} \text{card } A$
 $\langle \text{proof} \rangle$

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

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* \Rightarrow *'a mformula set* **where**
SUB (*Conj* φ ψ) = {*Conj* φ ψ } \cup *SUB* φ \cup *SUB* ψ
| *SUB* (*Disj* φ ψ) = {*Disj* φ ψ } \cup *SUB* φ \cup *SUB* ψ
| *SUB* (*Var* x) = {*Var* x }
| *SUB* *FALSE* = {*FALSE*}
| *SUB* *TRUE* = {*TRUE*}

the variables of a mformula

fun *vars* :: *'a mformula* \Rightarrow *'a set* **where**
vars (*Var* x) = { x }
| *vars* (*Conj* φ ψ) = *vars* φ \cup *vars* ψ
| *vars* (*Disj* φ ψ) = *vars* φ \cup *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* \Rightarrow *nat* **where**
cs φ = *card* (*SUB* φ)

variable assignments

type-synonym *'a VAS* = *'a* \Rightarrow *bool*

evaluation of mformulas

fun *eval* :: *'a VAS* \Rightarrow *'a mformula* \Rightarrow *bool* **where**
eval ϑ *FALSE* = *False*
| *eval* ϑ *TRUE* = *True*
| *eval* ϑ (*Var* x) = ϑ x
| *eval* ϑ (*Disj* φ ψ) = (*eval* ϑ φ \vee *eval* ϑ ψ)
| *eval* ϑ (*Conj* φ ψ) = (*eval* ϑ φ \wedge *eval* ϑ ψ)

lemma *eval-vars*: **assumes** $\bigwedge x. x \in \text{vars } \varphi \implies \vartheta 1 x = \vartheta 2 x$
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 ∈ tf-mformula
  | tf-Var: Var x ∈ tf-mformula
  | tf-Disj:  $\varphi \in \text{tf-mformula} \implies \psi \in \text{tf-mformula} \implies \text{Disj } \varphi \ \psi \in \text{tf-mformula}$ 
  | tf-Conj:  $\varphi \in \text{tf-mformula} \implies \psi \in \text{tf-mformula} \implies \text{Conj } \varphi \ \psi \in \text{tf-mformula}$ 

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 ∅ (to-tf-formula  $\varphi$ ) = eval ∅  $\varphi$ 
  ⟨proof⟩

lemma to-tf-formula: to-tf-formula  $\varphi \neq \text{TRUE} \implies \text{to-tf-formula } \varphi \in \text{tf-mformula}$ 
  ⟨proof⟩

lemma vars-to-tf-formula: vars (to-tf-formula  $\varphi$ ) ⊆ vars  $\varphi$ 
  ⟨proof⟩

lemma SUB-to-tf-formula: SUB (to-tf-formula  $\varphi$ ) ⊆ to-tf-formula ‘SUB  $\varphi$ 
  ⟨proof⟩

lemma cs-to-tf-formula: cs (to-tf-formula  $\varphi$ ) ≤ cs  $\varphi$ 
  ⟨proof⟩

lemma to-tf-mformula: assumes ¬ eval ∅  $\varphi$ 
  shows ∃  $\psi \in \text{tf-mformula}$ . (∀ ∅. eval ∅  $\varphi = \text{eval } \emptyset \ \psi$ ) ∧ vars  $\psi \subseteq \text{vars } \varphi \wedge \text{cs } \psi \leq \text{cs } \varphi$ 
  ⟨proof⟩

end

```

4 Simplified 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  $l * (p - 1)^l$ 

lemma kml:  $k \leq m - l$ 
<proof>
end

locale second-assumptions = first-assumptions +
  assumes kl2:  $k = l^2$ 
  and l8:  $l \geq 8$ 
begin

lemma Lm:  $L \geq m$ 
<proof>

lemma Lp:  $L > p$  <proof>

lemma L3:  $L > 3$  <proof>
end

definition eps =  $1/(1000 :: \text{real})$ 
lemma eps:  $eps > 0$  <proof>

definition L0 :: nat where
  L0 = (SOME l0.  $\forall l \geq l0. 1 / 3 < (1 + - 1 / \text{real } l)^l$ )

```

definition $M0 :: nat$ **where**

$M0 = (SOME\ y.\ \forall\ x.\ x \geq y \longrightarrow (\text{root } 8\ (\text{real } x) * \log 2\ (\text{real } x) + 1) / \text{real } x \text{ powr } (1 / 8 + \text{eps}) \leq 1)$

definition $L0' :: nat$ **where**

$L0' = (SOME\ l0.\ \forall\ n \geq l0.\ 6 * (\text{real } n)^{\wedge} 16 * \text{fact } n < \text{real } (n^2 \wedge 4) \text{ powr } (1 / 8 * \text{real } (n^2 \wedge 4) \text{ powr } (1 / 8)))$

definition $L0'' :: nat$ **where** $L0'' = (SOME\ l0.\ \forall\ l \geq l0.\ \text{real } l * \log 2\ (\text{real } (l^2 \wedge 4)) + 1 < \text{real } (l^2))$

lemma $L0''$: **assumes** $l \geq L0''$ **shows** $\text{real } l * \log 2\ (\text{real } (l^2 \wedge 4)) + 1 < \text{real } (l^2)$
{proof}

definition $M0' :: nat$ **where**

$M0' = (SOME\ x0.\ \forall\ x \geq x0.\ \text{real } x \text{ powr } (2 / 3) \leq x \text{ powr } (3 / 4) - 1)$

locale $\text{third-assumptions} = \text{second-assumptions} +$

assumes $\text{pllog}: l * \log 2\ m \leq p \text{ real } p \leq l * \log 2\ m + 1$

and $L0: l \geq L0$

and $L0': l \geq L0'$

and $M0': m \geq M0'$

and $M0: m \geq M0$

begin

lemma approximation1 :

$(\text{real } (k - 1))^{\wedge} (m - l) * \text{prod } (\lambda\ i.\ \text{real } (k - 1 - i)) \{0..<l\}$
 $> (\text{real } (k - 1))^{\wedge} m / 3$
{proof}

lemma approximation2 : **fixes** $s :: nat$

assumes $m \text{ choose } k \leq s * L^2 * (m - l - 1 \text{ choose } (k - l - 1))$

shows $((m - l) / k)^{\wedge} l / (6 * L^{\wedge} 2) < s$

{proof}

lemma approximation3 : **fixes** $s :: nat$

assumes $(k - 1)^{\wedge} m / 3 < (s * (L^2 * (k - 1)^{\wedge} m)) / 2^{\wedge} (p - 1)$

shows $((m - l) / k)^{\wedge} l / (6 * L^{\wedge} 2) < s$

{proof}

lemma identities : $k = \text{root } 4\ m\ l = \text{root } 8\ m$

{proof}

lemma identities2 : $\text{root } 4\ m = m \text{ powr } (1/4)\ \text{root } 8\ m = m \text{ powr } (1/8)$

{proof}

lemma appendix-A-1 : **assumes** $x \geq M0'$ **shows** $x \text{ powr } (2/3) \leq x \text{ powr } (3/4) -$

1
<proof>

lemma *appendix-A-2*: $(p - 1)^{\lceil l \rceil} < m \text{ powr } ((1 / 8 + \text{eps}) * l)$
<proof>

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

lemma *appendix-A-4*: $12 * L^2 \leq m \text{ powr } (m \text{ powr } (1 / 8) * 0.51)$
<proof>

lemma *approximation4*: **fixes** $s :: \text{nat}$
 assumes $s > ((m - l) / k)^{\lceil l \rceil} / (6 * L^2)$
 shows $s > 2 * k \text{ powr } (4 / 7 * \text{sqrt } k)$
<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* :: $\text{args} \Rightarrow 'a \text{ list } ([[(-]])$

no-syntax *--listcompr* :: $\text{args} \Rightarrow 'a \text{ list } ([[(-]])$

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

4.2 Plain Graphs

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

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

lemma *sameprod-altdef*: $X^{\mathbf{2}} = \{Y. Y \subseteq X \wedge \text{card } Y = 2\}$
<proof>

definition *numbers* :: $\text{nat} \Rightarrow \text{nat set}$ $([[(-]])$ **where**
 $[n] \equiv \{\cdot < n\}$

lemma *card-sameprod*: $\text{finite } X \implies \text{card } (X^{\mathbf{2}}) = \text{card } X \text{ choose } 2$
<proof>

lemma *sameprod-mono*: $X \subseteq Y \implies X^{\mathbf{2}} \subseteq Y^{\mathbf{2}}$
<proof>

lemma *sameprod-finite*: $\text{finite } X \implies \text{finite } (X^{\mathbf{2}})$
<proof>

lemma *numbers2-mono*: $x \leq y \implies [x]^{\mathbf{2}} \subseteq [y]^{\mathbf{2}}$
<proof>

lemma *card-numbers[simp]*: $\text{card } [n] = n$
<proof>

lemma *card-numbers2[simp]*: $\text{card } ([n]^{\mathbf{2}}) = n \text{ choose } 2$
<proof>

type-synonym *vertex* = *nat*

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

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

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

context *first-assumptions*

begin

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

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

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

definition *v* :: *graph* \Rightarrow *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**
 $\mathcal{K} = \{ K . K \in \mathcal{G} \wedge \text{card } (v K) = k \wedge K = (v K)^{\mathbf{2}} \}$

lemma *v-G*: $G \in \mathcal{G} \implies v G \subseteq [m]$
<proof>

lemma *v-mono*: $G \subseteq H \implies v G \subseteq v H$ *<proof>*

lemma *v-sameprod*[*simp*]: **assumes** $\text{card } X \geq 2$
shows $v (X^{\mathbf{2}}) = X$
<proof>

lemma *v-mem-sub*: **assumes** $\text{card } e = 2$ $e \in G$ **shows** $e \subseteq v G$
<proof>

lemma *v-G-2*: **assumes** $G \in \mathcal{G}$ **shows** $G \subseteq (v G)^{\mathbf{2}}$
<proof>

lemma *v-numbers2*[*simp*]: $x \geq 2 \implies v ([x]^{\mathbf{2}}) = [x]$
<proof>

lemma *sameprod-G*: **assumes** $X \subseteq [m]$ $\text{card } X \geq 2$
shows $X^{\mathbf{2}} \in \mathcal{G}$
<proof>

lemma *finite-numbers*[*simp,intro*]: *finite* $[n]$
<proof>

lemma *finite-numbers2*[*simp,intro*]: *finite* $([n]^{\mathbf{2}})$
<proof>

lemma *finite-members-G*: $G \in \mathcal{G} \implies \text{finite } G$
<proof>

lemma *finite-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 $2 \leq \text{card } (v G)$
<proof>

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

lemma *K-G*: $\mathcal{K} \subseteq \mathcal{G}$
<proof>

definition *CLIQUE* :: graph set **where**
 $CLIQUE = \{ G. G \in \mathcal{G} \wedge (\exists K \in \mathcal{K}. K \subseteq G) \}$

lemma *empty-CLIQUE[simp]*: $\{\} \notin CLIQUE$ *<proof>*

4.3 Test Graphs

Positive test graphs are precisely the cliques of size k .

abbreviation $POS \equiv \mathcal{K}$

lemma *POS-G*: $POS \subseteq \mathcal{G}$ *<proof>*

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

type-synonym $colorf = vertex \Rightarrow nat$

definition \mathcal{F} :: colorf set **where**
 $\mathcal{F} = [m] \rightarrow_E [k - 1]$

lemma *finite-F*: finite \mathcal{F}
<proof>

definition \mathcal{C} :: colorf \Rightarrow graph **where**
 $\mathcal{C} f = \{ \{x, y\} \mid x \neq y \wedge \{x, y\} \in [m] \times [m] \wedge f x \neq f y \}$

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

Lemma 1 **lemma** *CLIQUE-NEG*: $CLIQUE \cap NEG = \{\}$
<proof>

lemma *NEG-G*: $NEG \subseteq \mathcal{G}$
<proof>

lemma *finite-POS-NEG*: finite $(POS \cup NEG)$
<proof>

lemma *POS-sub-CLIQUE*: $POS \subseteq CLIQUE$
<proof>

lemma *POS-CLIQUE*: $POS \subset CLIQUE$
<proof>

lemma *card-POS*: card $POS = m$ choose k
<proof>

4.4 Basic operations on sets of graphs

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

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

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

4.5 Acceptability

Definition 2

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

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

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

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

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

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

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

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

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

Lemma 3

lemma *ACC-cf-empty*: $ACC\text{-}cf\ \{\} = \{\}$
 \langle *proof* \rangle

lemma *ACC-empty*[*simp*]: $ACC\ \{\} = \{\}$
 \langle *proof* \rangle

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

definition *v-gs* :: graph set \Rightarrow 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*: $X \subseteq Y \implies v\text{-}gs X \subseteq v\text{-}gs Y$
 ⟨proof⟩

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

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

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

definition *odotl* :: graph set \Rightarrow graph set \Rightarrow graph set (*infixl* $\odot l$ 65) where
 $X \odot l Y = (X \odot Y) \cap Gl$

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 $\text{card } (v\text{-gs } Z) \leq \text{card } (v\text{-gs } X) * \text{card } (v\text{-gs } Y)$
 $\langle \text{proof} \rangle$

Definition 6 – elementary plucking step

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

context *second-assumptions*
begin

Lemma 9 – for elementary plucking step

lemma *v-sameprod-subset*: $v (Vs^{\wedge 2}) \subseteq Vs$ $\langle \text{proof} \rangle$

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

Definition 6

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

termination
 $\langle \text{proof} \rangle$

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

definition *PLU* :: *graph set* \Rightarrow *graph set* **where**
 $PLU X = \text{fst } (PLU\text{-main } X)$

Lemma 7

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

<proof>

Definition 8

definition *sqcup* :: graph set \Rightarrow graph set \Rightarrow graph set (**infixl** \sqcup 65) **where**
 $X \sqcup Y = PLU (X \cup Y)$

definition *sqcap* :: graph set \Rightarrow graph set \Rightarrow graph set (**infixl** \sqcap 65) **where**
 $X \sqcap Y = PLU (X \odot l Y)$

definition *deviate-pos-cup* :: graph set \Rightarrow graph set \Rightarrow graph set ($\partial \sqcup Pos$) **where**
 $\partial \sqcup Pos X Y = POS \cap ACC (X \cup Y) - ACC (X \sqcup Y)$

definition *deviate-pos-cap* :: graph set \Rightarrow graph set \Rightarrow graph set ($\partial \sqcap Pos$) **where**
 $\partial \sqcap Pos X Y = POS \cap ACC (X \odot Y) - ACC (X \sqcap Y)$

definition *deviate-neg-cup* :: graph set \Rightarrow graph set \Rightarrow colorf set ($\partial \sqcup Neg$) **where**
 $\partial \sqcup Neg X Y = ACC\text{-}cf (X \sqcup Y) - ACC\text{-}cf (X \cup Y)$

definition *deviate-neg-cap* :: graph set \Rightarrow graph set \Rightarrow colorf set ($\partial \sqcap Neg$) **where**
 $\partial \sqcap Neg X Y = ACC\text{-}cf (X \sqcap Y) - ACC\text{-}cf (X \odot Y)$

Lemma 9 – without applying Lemma 7

lemma *PLU-main*: **assumes** $X \subseteq Gl$

and *PLU-main* $X = (Z, n)$

shows $Z \in \mathcal{PLGl}$

$\wedge (Z = \{\} \longleftrightarrow X = \{\})$

$\wedge POS \cap ACC X \subseteq ACC Z$

$\wedge 2^{\wedge p} * \text{card} (ACC\text{-}cf Z - ACC\text{-}cf X) \leq (k - 1)^{\wedge m} * n$

<proof>

Lemma 9

lemma **assumes** $X: X \in \mathcal{PLGl}$ **and** $Y: Y \in \mathcal{PLGl}$

shows *PLU-union*: $PLU (X \cup Y) \in \mathcal{PLGl}$ **and**

sqcup: $X \sqcup Y \in \mathcal{PLGl}$ **and**

sqcup-sub: $POS \cap ACC (X \cup Y) \subseteq ACC (X \sqcup Y)$ **and**

deviate-pos-cup: $\partial \sqcup Pos X Y = \{\}$ **and**

deviate-neg-cup: $\text{card} (\partial \sqcup Neg X Y) < (k - 1)^{\wedge m} * L / 2^{\wedge (p - 1)}$

<proof>

Lemma 10

lemma **assumes** $X: X \in \mathcal{PLGl}$ **and** $Y: Y \in \mathcal{PLGl}$

shows *PLU-joinl*: $PLU (X \odot l Y) \in \mathcal{PLGl}$ **and**

sqcap: $X \sqcap Y \in \mathcal{PLGl}$ **and**

deviate-neg-cap: $\text{card} (\partial \sqcap Neg X Y) < (k - 1)^{\wedge m} * L^{\wedge 2} / 2^{\wedge (p - 1)}$ **and**

deviate-pos-cap: $\text{card} (\partial \sqcap Pos X Y) \leq ((m - l - 1) \text{ choose } (k - l - 1)) * L^{\wedge 2}$

<proof>

end

4.7 Formalism

Fix a variable set of cardinality m over 2 .

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

definition n **where** $n = (m \text{ choose } 2)$

the formulas over the fixed variable set

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

lemma *\mathcal{A} -simps*[*simp*]:

$FALSE \in \mathcal{A}$
 $(\text{Var } x \in \mathcal{A}) = (x \in \mathcal{V})$
 $(\text{Conj } \varphi \ \psi \in \mathcal{A}) = (\varphi \in \mathcal{A} \wedge \psi \in \mathcal{A})$
 $(\text{Disj } \varphi \ \psi \in \mathcal{A}) = (\varphi \in \mathcal{A} \vee \psi \in \mathcal{A})$
 $\langle \text{proof} \rangle$

lemma *inj-on- π* : *inj-on* $\pi \ \mathcal{V}$

$\langle \text{proof} \rangle$

lemma $\pi m2$ [*simp,intro*]: $x \in \mathcal{V} \Longrightarrow \pi \ x \in [m] \hat{=} 2$

$\langle \text{proof} \rangle$

lemma *card-v- π* [*simp,intro*]: **assumes** $x \in \mathcal{V}$

shows $\text{card } (v \ \{\pi \ x\}) = 2$

$\langle \text{proof} \rangle$

lemma *π -singleton*[*simp,intro*]: **assumes** $x \in \mathcal{V}$

shows $\{\pi \ x\} \in \mathcal{G}$

$\{\{\pi \ x\}\} \in \text{PLGl}$

$\langle \text{proof} \rangle$

lemma *empty-PLGl*[*simp,intro*]: $\{\} \in \text{PLGl}$

$\langle \text{proof} \rangle$

fun *SET* :: *'a mformula* \Rightarrow *graph set* **where**

$SET \ FALSE = \{\}$

| $SET \ (\text{Var } x) = \{\{\pi \ x\}\}$

| $SET \ (\text{Disj } \varphi \ \psi) = SET \ \varphi \cup SET \ \psi$

| $SET \ (\text{Conj } \varphi \ \psi) = SET \ \varphi \odot SET \ \psi$

lemma *ACC-cf-SET*[*simp*]:

$ACC\text{-cf} \ (SET \ (\text{Var } x)) = \{f \in \mathcal{F}. \pi \ x \in Cf\}$

$ACC\text{-cf} \ (SET \ FALSE) = \{\}$

$$\begin{aligned}
ACC\text{-}cf (SET (Disj \varphi \psi)) &= ACC\text{-}cf (SET \varphi) \cup ACC\text{-}cf (SET \psi) \\
ACC\text{-}cf (SET (Conj \varphi \psi)) &= ACC\text{-}cf (SET \varphi) \cap ACC\text{-}cf (SET \psi) \\
\langle proof \rangle
\end{aligned}$$

lemma *ACC-SET[simp]*:

$$\begin{aligned}
ACC (SET (Var x)) &= \{G \in \mathcal{G}. \pi x \in G\} \\
ACC (SET FALSE) &= \{\} \\
ACC (SET (Disj \varphi \psi)) &= ACC (SET \varphi) \cup ACC (SET \psi) \\
ACC (SET (Conj \varphi \psi)) &= ACC (SET \varphi) \cap ACC (SET \psi) \\
\langle proof \rangle
\end{aligned}$$

lemma *SET-G*: $\varphi \in \text{tf-mformula} \implies \varphi \in \mathcal{A} \implies SET \varphi \subseteq \mathcal{G}$
 $\langle proof \rangle$

fun *APR* :: 'a mformula \Rightarrow graph set **where**

$$\begin{aligned}
APR FALSE &= \{\} \\
| APR (Var x) &= \{\{\pi x\}\} \\
| APR (Disj \varphi \psi) &= APR \varphi \sqcup APR \psi \\
| APR (Conj \varphi \psi) &= APR \varphi \sqcap APR \psi
\end{aligned}$$

lemma *APR*: $\varphi \in \text{tf-mformula} \implies \varphi \in \mathcal{A} \implies APR \varphi \in \mathcal{PLGI}$
 $\langle proof \rangle$

definition *ACC-cf-mf* :: 'a mformula \Rightarrow colorf set **where**

$$ACC\text{-}cf\text{-}mf \varphi = ACC\text{-}cf (SET \varphi)$$

definition *ACC-mf* :: 'a mformula \Rightarrow graph set **where**

$$ACC\text{-}mf \varphi = ACC (SET \varphi)$$

definition *deviate-pos* :: 'a mformula \Rightarrow graph set (∂Pos) **where**

$$\partial Pos \varphi = POS \cap ACC\text{-}mf \varphi - ACC (APR \varphi)$$

definition *deviate-neg* :: 'a mformula \Rightarrow colorf set (∂Neg) **where**

$$\partial Neg \varphi = ACC\text{-}cf (APR \varphi) - ACC\text{-}cf\text{-}mf \varphi$$

Lemma 11.1

lemma *deviate-subset-Disj*:

$$\begin{aligned}
\partial Pos (Disj \varphi \psi) &\subseteq \partial \sqcup Pos (APR \varphi) (APR \psi) \cup \partial Pos \varphi \cup \partial Pos \psi \\
\partial Neg (Disj \varphi \psi) &\subseteq \partial \sqcup Neg (APR \varphi) (APR \psi) \cup \partial Neg \varphi \cup \partial Neg \psi \\
\langle proof \rangle
\end{aligned}$$

Lemma 11.2

lemma *deviate-subset-Conj*:

$$\begin{aligned}
\partial Pos (Conj \varphi \psi) &\subseteq \partial \sqcap Pos (APR \varphi) (APR \psi) \cup \partial Pos \varphi \cup \partial Pos \psi \\
\partial Neg (Conj \varphi \psi) &\subseteq \partial \sqcap Neg (APR \varphi) (APR \psi) \cup \partial Neg \varphi \cup \partial Neg \psi \\
\langle proof \rangle
\end{aligned}$$

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

lemma *deviate-finite*:

finite ($\partial Pos \varphi$)
finite ($\partial Neg \varphi$)
finite ($\partial \sqcup Pos A B$)
finite ($\partial \sqcup Neg A B$)
finite ($\partial \sqcap Pos A B$)
finite ($\partial \sqcap Neg A B$)
 $\langle proof \rangle$

Lemma 12

lemma *no-deviation[simp]*:

$\partial Pos FALSE = \{\}$
 $\partial Neg FALSE = \{\}$
 $\partial Pos (Var x) = \{\}$
 $\partial Neg (Var x) = \{\}$
 $\langle proof \rangle$

Lemma 12.1-2

fun *approx-pos* **where**

approx-pos (*Conj phi psi*) = $\partial \sqcap Pos (APR phi) (APR psi)$
| *approx-pos* - = $\{\}$

fun *approx-neg* **where**

approx-neg (*Conj phi psi*) = $\partial \sqcap Neg (APR phi) (APR psi)$
| *approx-neg* (*Disj phi psi*) = $\partial \sqcup Neg (APR phi) (APR psi)$
| *approx-neg* - = $\{\}$

lemma *finite-approx-pos*: *finite* (*approx-pos* φ)

$\langle proof \rangle$

lemma *finite-approx-neg*: *finite* (*approx-neg* φ)

$\langle proof \rangle$

lemma *card-deviate-Pos*: **assumes** *phi*: $\varphi \in \text{tf-mformula}$ $\varphi \in \mathcal{A}$

shows $\text{card} (\partial Pos \varphi) \leq cs \varphi * L^2 * ((m - l - 1) \text{choose} (k - l - 1))$

$\langle proof \rangle$

lemma *card-deviate-Neg*: **assumes** *phi*: $\varphi \in \text{tf-mformula}$ $\varphi \in \mathcal{A}$

shows $\text{card} (\partial Neg \varphi) \leq cs \varphi * L^2 * (k - 1)^m / 2^{p-1}$

$\langle proof \rangle$

Lemma 12.3

lemma *ACC-cf-non-empty-approx*: **assumes** *phi*: $\varphi \in \text{tf-mformula}$ $\varphi \in \mathcal{A}$

and *ne*: $APR \varphi \neq \{\}$

shows $\text{card} (ACC\text{-cf} (APR \varphi)) > (k - 1)^m / 3$

$\langle proof \rangle$

Theorem 13

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

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

Definition 14

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

definition *eval-gs* :: 'a VAS \Rightarrow graph set \Rightarrow 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*:
 $eval\text{-gs } \vartheta (X \cup Y) = (eval\text{-gs } \vartheta X \vee eval\text{-gs } \vartheta Y)$
 $\langle \text{proof} \rangle$

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

Lemma 15

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

definition ϑ_g :: graph \Rightarrow '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*: $\varphi \in \text{tf-mformula } \varphi \in \mathcal{A}$
and *G*: $G \in \mathcal{G}$
shows $eval (\vartheta_g G) \varphi = (G \in ACC\text{-mf } \varphi)$
 $\langle \text{proof} \rangle$

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*: $\varphi \in \text{tf-mformula}$
and *phi*: $\varphi \in \mathcal{A}$
shows $POS \subseteq ACC\text{-mf } \varphi$
 $\langle \text{proof} \rangle$

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*: $\varphi \in \text{tf-mformula}$
and *phi*: $\varphi \in \mathcal{A}$
shows $ACC\text{-cf-mf } \varphi = \{\}$

<proof>

4.8 Conclusion

Theorem 22

We first consider monotone formulas without TRUE.

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

Next we consider general monotone formulas.

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

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

Let us now move the theorem outside the locale

definition *Large-Number* **where** *Large-Number* = $\text{Max } \{64, L0'' \wedge 2, L0 \wedge 2, L0' \wedge 2, M0, M0'\}$

theorem *Clique-not-solvable-by-small-monotone-circuit-squared*:

fixes $\varphi :: 'a \text{ mformula}$
assumes *k*: $\exists l. k = l \wedge 2$
and *LARGE*: $k \geq \text{Large-Number}$
and π : *bij-betw* $\pi V [k \wedge 4] \wedge 2$
and *solution*: $\forall G \in \text{Graphs } [k \wedge 4]. (G \in \text{Clique } [k \wedge 4] k) = \text{eval} (\lambda x. \pi x \in G) \varphi$
and *vars*: $\text{vars } \varphi \subseteq V$
shows $cs \varphi > k \text{ powr } (4 / 7 * \text{sqrt } k)$
<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 $\varphi :: 'a\ mformula$
assumes $LARGE: k \geq Large-Number$
and $\pi: bij\text{-}betw\ \pi\ V\ [k^{\wedge}8]^{\wedge}2$
and $solution: \forall G \in Graphs\ [k^{\wedge}8]. (G \in Clique\ [k^{\wedge}8]\ (k^{\wedge}2)) = eval\ (\lambda x. \pi\ x\ x \in G)\ \varphi$
and $vars: vars\ \varphi \subseteq V$
shows $cs\ \varphi > k\ powr\ (8 / 7 * k)$
 $\langle proof \rangle$

definition *large-number* **where** $large\text{-}number = Large-Number^{\wedge}8$

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 $\varphi :: 'a\ mformula$
assumes $large: n \geq large\text{-}number$
and $kn: \exists k. n = k^{\wedge}8$
and $\pi: bij\text{-}betw\ \pi\ V\ [n]^{\wedge}2$
and $s: s = root\ 4\ n$
and $solution: \forall G \in Graphs\ [n]. (G \in Clique\ [n]\ s) = eval\ (\lambda x. \pi\ x\ x \in G)\ \varphi$
and $vars: vars\ \varphi \subseteq V$
shows $cs\ \varphi > (root\ 7\ n)\ powr\ (root\ 8\ n)$
 $\langle proof \rangle$

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. Available at <http://arxiv.org/abs/2005.00809v3>.
- [5] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.