

# 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  $\sqrt[n]{n}^{\frac{8}{\sqrt[n]{n}}}$  for the fixed choice of  $s = \sqrt[n]{n}$ ), following a proof by Gordeev.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Preliminaries</b>	<b>2</b>
<b>3</b>	<b>Monotone Formulas</b>	<b>7</b>
3.1	Definition . . . . .	7
3.2	Conversion of mformulas to true-free mformulas . . . . .	8
<b>4</b>	<b>Simplified Version of Gordeev’s Proof for Monotone Circuits</b>	<b>9</b>
4.1	Setup of Global Assumptions and Proofs of Approximations .	9
4.2	Plain Graphs . . . . .	19
4.3	Test Graphs . . . . .	23
4.4	Basic operations on sets of graphs . . . . .	26
4.5	Acceptability . . . . .	26
4.6	Approximations and deviations . . . . .	28
4.7	Formalism . . . . .	45
4.8	Conclusion . . . . .	56

---

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We arrived at this formalization by trying to verify an unpublished draft of Gordeev [4], which tries to show that Clique cannot be solved by any polynomial-sized circuit, including non-monotone ones, where the concrete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

**using** *assms*

**proof** (*induct n arbitrary: A C*)

**case** *0*

**thus** *?case using finite-subset[of A C] by (intro exI[of - {}], auto)*

**next**

```

case (Suc n A C)
show ?case
proof (cases A = {})
  case True
  from obtain-subset-with-card-n[OF Suc(3)]
  obtain B where B ⊆ C card B = Suc n by metis
  thus ?thesis unfolding True by blast
next
  case False
  then obtain a where a: a ∈ A by auto
  let ?A = A - {a}
  let ?C = C - {a}
  have 1: card ?A ≤ n using Suc(2-) a
    using finite-subset by fastforce
  have 2: card ?C ≥ n using Suc(2-) a by auto
  from Suc(1)[OF 1 2 - finite-subset[OF - Suc(5)]] Suc(2-)
  obtain B where ?A ⊆ B B ⊆ ?C card B = n by blast
  thus ?thesis using a Suc(2-)
    by (intro exI[of - insert a B], auto intro!: card-insert-disjoint finite-subset[of
B C])
qed
qed

```

```

lemma fact-approx-add: fact (l + n) ≤ fact l * (real l + real n) ^ n
proof (induct n arbitrary: l)
  case (Suc n l)
  have fact (l + Suc n) = (real l + Suc n) * fact (l + n) by simp
  also have ... ≤ (real l + Suc n) * (fact l * (real l + real n) ^ n)
    by (intro mult-left-mono[OF Suc], auto)
  also have ... = fact l * ((real l + Suc n) * (real l + real n) ^ n) by simp
  also have ... ≤ fact l * ((real l + Suc n) * (real l + real (Suc n)) ^ n)
    by (rule mult-left-mono, rule mult-left-mono, rule power-mono, auto)
  finally show ?case by simp
qed simp

```

```

lemma fact-approx-minus: assumes k ≥ n
  shows fact k ≤ fact (k - n) * (real k ^ n)
proof -
  define l where l = k - n
  from assms have k: k = l + n unfolding l-def by auto
  show ?thesis unfolding k using fact-approx-add[of l n] by simp
qed

```

```

lemma fact-approx-upper-add: assumes a1: a ≤ Suc l shows fact l * real a ^ n
≤ fact (l + n)
proof (induct n)
  case (Suc n)
  have fact l * real a ^ (Suc n) = (fact l * real a ^ n) * real a by simp
  also have ... ≤ fact (l + n) * real a

```

by (rule mult-right-mono[OF Suc], auto)  
 also have ...  $\leq$  fact (l + n) \* real (Suc (l + n))  
 by (intro mult-left-mono, insert al, auto)  
 also have ... = fact (Suc (l + n)) by simp  
 finally show ?case by simp  
 qed simp

**lemma fact-approx-upper-minus:** assumes  $n \leq k$  and  $n + a \leq \text{Suc } k$   
 shows fact (k - n) \* real  $a^n \leq$  fact k  
**proof** -  
 define l where  $l = k - n$   
 from assms have  $k: k = l + n$  unfolding l-def by auto  
 show ?thesis using assms unfolding k  
 apply simp  
 apply (rule fact-approx-upper-add, insert assms, auto simp: l-def)  
 done  
 qed

**lemma choose-mono:**  $n \leq m \implies n \text{ choose } k \leq m \text{ choose } k$   
 unfolding binomial-def  
 by (rule card-mono, auto)

**lemma div-mult-le:**  $(a \text{ div } b) * c \leq (a * c) \text{ div } (b :: \text{nat})$   
 by (metis div-mult2-eq div-mult-mult2 mult.commute mult-0-right times-div-less-eq-dividend)

**lemma div-mult-pow-le:**  $(a \text{ div } b)^n \leq a^n \text{ div } (b :: \text{nat})^n$   
**proof** (cases  $b = 0$ )  
 case True  
 thus ?thesis by (cases n, auto)  
 next  
 case b: False  
 then obtain c d where  $a: a = b * c + d$  and  $id: c = a \text{ div } b$   $d = a \text{ mod } b$  by  
 auto  
 have  $(a \text{ div } b)^n = c^n$  unfolding id by simp  
 also have ... =  $(b * c)^n \text{ div } b^n$  using b  
 by (metis div-power dvd-triv-left nonzero-mult-div-cancel-left)  
 also have ...  $\leq (b * c + d)^n \text{ div } b^n$   
 by (rule div-le-mono, rule power-mono, auto)  
 also have ... =  $a^n \text{ div } b^n$  unfolding a by simp  
 finally show ?thesis .  
 qed

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

```

define  $m$  where  $m = \min n k$ 
define  $M$  where  $M = \max n k$ 
from  $nk$  have  $mM$ :  $m < M$  unfolding  $m$ -def  $M$ -def by auto
let  $?new = \text{insert } (M - 1) \{0..<l - 1\}$ 
let  $?m = \{K \in \text{Pow } \{0..<m\}. \text{card } K = l\}$ 
let  $?M = \{K \in \text{Pow } \{0..<M\}. \text{card } K = l\}$ 
from  $id\ n0$  have  $lM$  :  $l \leq M$  unfolding  $m$ -def  $M$ -def by auto
from  $id$  have  $id$ :  $(m \text{ choose } l) = (M \text{ choose } l)$ 
  unfolding  $m$ -def  $M$ -def by auto
from  $this$ [unfolded binomial-def]
have  $\text{card } ?M < \text{Suc } (\text{card } ?m)$ 
  by auto
also have  $\dots = \text{card } (\text{insert } ?new ?m)$ 
  by (rule sym, rule card-insert-disjoint, force, insert mM, auto)
also have  $\dots \leq \text{card } (\text{insert } ?new ?M)$ 
  by (rule card-mono, insert mM, auto)
also have  $\text{insert } ?new ?M = ?M$ 
  by (insert mM lM l0, auto)
finally show False by simp
qed

```

**lemma** *card-funcsetE*:  $\text{finite } A \implies \text{card } (A \rightarrow_E B) = \text{card } B \wedge \text{card } A$   
**by** (*subst card-PiE, auto*)

**lemma** *card-inj-on-subset-funcset*: **assumes**  $\text{finB}$ :  $\text{finite } B$   
**and**  $\text{finC}$ :  $\text{finite } C$   
**and**  $AB$ :  $A \subseteq B$

**shows**  $\text{card } \{f. f \in B \rightarrow_E C \wedge \text{inj-on } f A\} =$   
 $\text{card } C \wedge (\text{card } B - \text{card } A) * \text{prod } ((-) (\text{card } C)) \{0 ..< \text{card } A\}$

**proof** –

**define**  $D$  **where**  $D = B - A$

**from**  $AB$  **have**  $B$ :  $B = A \cup D$  **and**  $\text{disj}$ :  $A \cap D = \{\}$  **unfolding**  $D$ -def **by** *auto*

**have**  $\text{sub}$ :  $\text{card } B - \text{card } A = \text{card } D$  **unfolding**  $D$ -def **using**  $\text{finB } AB$

**by** (*metis card-Diff-subset finite-subset*)

**have**  $\text{finite } A$   $\text{finite } D$  **using**  $\text{finB}$  **unfolding**  $B$  **by** *auto*

**thus**  $?thesis$  **unfolding**  $\text{sub}$  **unfolding**  $B$  **using**  $\text{disj}$

**proof** (*induct A rule: finite-induct*)

**case** *empty*

**from**  $\text{card-funcsetE}$ [*OF this(1), of C*] **show**  $?case$  **by** *auto*

**next**

**case** (*insert a A*)

**have**  $\{f. f \in \text{insert } a A \cup D \rightarrow_E C \wedge \text{inj-on } f (\text{insert } a A)\}$

$= \{f(a := c) \mid f c. f \in A \cup D \rightarrow_E C \wedge \text{inj-on } f A \wedge c \in C - f ' A\}$

(**is**  $?l = ?r$ )

**proof**

**show**  $?r \subseteq ?l$

**by** (*auto intro: inj-on-fun-updI split: if-splits*)

{

**fix**  $f$

```

assume  $f: f \in ?l$ 
let  $?g = f(a := \text{undefined})$ 
let  $?h = ?g(a := f a)$ 
have  $\text{mem}: f a \in C - ?g \text{ ' } A$  using  $\text{insert}(1,2,4,5) f$  by  $\text{auto}$ 
from  $f$  have  $f: f \in \text{insert } a \ A \cup D \rightarrow_E C$   $\text{inj-on } f \ (\text{insert } a \ A)$  by  $\text{auto}$ 
hence  $?g \in A \cup D \rightarrow_E C$   $\text{inj-on } ?g \ A$  using  $\langle a \notin A \rangle \langle \text{insert } a \ A \cap D = \{\} \rangle$ 
by  $(\text{auto split: if-splits simp: inj-on-def})$ 
with  $\text{mem}$  have  $?h \in ?r$  by  $\text{blast}$ 
also have  $?h = f$  by  $\text{auto}$ 
finally have  $f \in ?r$  .
}
thus  $?l \subseteq ?r$  by  $\text{auto}$ 
qed
also have  $\dots = (\lambda (f, c). f (a := c)) \text{ '}$ 
 $(\text{Sigma } \{f . f \in A \cup D \rightarrow_E C \wedge \text{inj-on } f \ A\} (\lambda f. C - f \text{ ' } A))$ 
by  $\text{auto}$ 
also have  $\text{card } (\dots) = \text{card } (\text{Sigma } \{f . f \in A \cup D \rightarrow_E C \wedge \text{inj-on } f \ A\} (\lambda f. C - f \text{ ' } A))$ 
proof  $(\text{rule card-image, intro inj-onI, clarsimp, goal-cases})$ 
case  $(1 f c g d)$ 
let  $?f = f(a := c, a := \text{undefined})$ 
let  $?g = g(a := d, a := \text{undefined})$ 
from  $1$  have  $\text{id}: f(a := c) = g(a := d)$  by  $\text{auto}$ 
from  $\text{fun-upd-eqD}[OF \text{id}]$ 
have  $\text{cd}: c = d$  by  $\text{auto}$ 
from  $\text{id}$  have  $?f = ?g$  by  $\text{auto}$ 
also have  $?f = f$  using  $\langle f \in A \cup D \rightarrow_E C \rangle \text{insert}(1,2,4,5)$ 
by  $(\text{intro ext, auto})$ 
also have  $?g = g$  using  $\langle g \in A \cup D \rightarrow_E C \rangle \text{insert}(1,2,4,5)$ 
by  $(\text{intro ext, auto})$ 
finally show  $f = g \wedge c = d$  using  $\text{cd}$  by  $\text{auto}$ 
qed
also have  $\dots = (\sum f \in \{f \in A \cup D \rightarrow_E C. \text{inj-on } f \ A\}. \text{card } (C - f \text{ ' } A))$ 
by  $(\text{rule card-SigmaI, rule finite-subset}[of - A \cup D \rightarrow_E C],$ 
 $\text{insert } \langle \text{finite } C \rangle \langle \text{finite } D \rangle \langle \text{finite } A \rangle, \text{auto intro!: finite-PiE})$ 
also have  $\dots = (\sum f \in \{f \in A \cup D \rightarrow_E C. \text{inj-on } f \ A\}. \text{card } C - \text{card } A)$ 
by  $(\text{rule sum.cong}[OF refl], \text{subst card-Diff-subset, insert } \langle \text{finite } A \rangle, \text{auto simp: card-image})$ 
also have  $\dots = (\text{card } C - \text{card } A) * \text{card } \{f \in A \cup D \rightarrow_E C. \text{inj-on } f \ A\}$ 
by  $\text{simp}$ 
also have  $\dots = \text{card } C \wedge \text{card } D * ((\text{card } C - \text{card } A) * \text{prod } ((-) (\text{card } C)) \{0..<\text{card } A\})$ 
using  $\text{insert}$  by  $(\text{auto simp: ac-simps})$ 
also have  $(\text{card } C - \text{card } A) * \text{prod } ((-) (\text{card } C)) \{0..<\text{card } A\} =$ 
 $\text{prod } ((-) (\text{card } C)) \{0..<\text{Suc } (\text{card } A)\}$  by  $\text{simp}$ 
also have  $\text{Suc } (\text{card } A) = \text{card } (\text{insert } a \ A)$  using  $\text{insert}$  by  $\text{auto}$ 
finally show  $?case$  .
qed
qed

```

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  $\varphi$   $\psi$ ) = {Conj  $\varphi$   $\psi$ }  $\cup$  SUB  $\varphi$   $\cup$  SUB  $\psi$ 
| SUB (Disj  $\varphi$   $\psi$ ) = {Disj  $\varphi$   $\psi$ }  $\cup$  SUB  $\varphi$   $\cup$  SUB  $\psi$ 
| 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  $\varphi$   $\psi$ ) = vars  $\varphi$   $\cup$  vars  $\psi$ 
| vars (Disj  $\varphi$   $\psi$ ) = vars  $\varphi$   $\cup$  vars  $\psi$ 
| vars FALSE = {}
| vars TRUE = {}
```

```
lemma finite-SUB[simp, intro]: finite (SUB  $\varphi$ )
by (induct  $\varphi$ , auto)
```

The circuit-size of a mformula: number of subformulas

```
definition cs :: 'a mformula  $\Rightarrow$  nat where
  cs  $\varphi$  = card (SUB  $\varphi$ )
```

variable assignments

```
type-synonym 'a VAS = 'a  $\Rightarrow$  bool
```

evaluation of mformulas

```

fun eval :: 'a VAS  $\Rightarrow$  'a mformula  $\Rightarrow$  bool where
  eval  $\vartheta$  FALSE = False
| eval  $\vartheta$  TRUE = True
| eval  $\vartheta$  (Var x) =  $\vartheta$  x
| eval  $\vartheta$  (Disj  $\varphi$   $\psi$ ) = (eval  $\vartheta$   $\varphi$   $\vee$  eval  $\vartheta$   $\psi$ )
| eval  $\vartheta$  (Conj  $\varphi$   $\psi$ ) = (eval  $\vartheta$   $\varphi$   $\wedge$  eval  $\vartheta$   $\psi$ )

```

```

lemma eval-vars: assumes  $\bigwedge x. x \in \text{vars } \varphi \implies \vartheta 1 x = \vartheta 2 x$ 
shows eval  $\vartheta 1$   $\varphi$  = eval  $\vartheta 2$   $\varphi$ 
using assms by (induct  $\varphi$ , auto)

```

### 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  $\implies \psi \in$  tf-mformula  $\implies$  Disj  $\varphi$   $\psi \in$  tf-mformula
| tf-Conj:  $\varphi \in$  tf-mformula  $\implies \psi \in$  tf-mformula  $\implies$  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  $\vee$  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  $\vartheta$  (to-tf-formula  $\varphi$ ) = eval  $\vartheta$   $\varphi$ 
by (induct  $\varphi$  rule: to-tf-formula.induct, auto simp: Let-def)

```

```

lemma to-tf-formula: to-tf-formula  $\varphi \neq$  TRUE  $\implies$  to-tf-formula  $\varphi \in$  tf-mformula

by (induct  $\varphi$ , auto simp: Let-def intro: tf-mformula.intros)

```

```

lemma vars-to-tf-formula: vars (to-tf-formula  $\varphi$ )  $\subseteq$  vars  $\varphi$ 
by (induct  $\varphi$  rule: to-tf-formula.induct, auto simp: Let-def)

```

```

lemma SUB-to-tf-formula: SUB (to-tf-formula  $\varphi$ )  $\subseteq$  to-tf-formula ' SUB  $\varphi$ 
by (induct  $\varphi$  rule: to-tf-formula.induct, auto simp: Let-def)

```

```

lemma cs-to-tf-formula: cs (to-tf-formula  $\varphi$ )  $\leq$  cs  $\varphi$ 

```

**proof** –

```

have cs (to-tf-formula  $\varphi$ )  $\leq$  card (to-tf-formula ' SUB  $\varphi$ )
  unfolding cs-def by (rule card-mono[OF finite-imageI[OF finite-SUB] SUB-to-tf-formula])
also have ...  $\leq$  cs  $\varphi$  unfolding cs-def
  by (rule card-image-le[OF finite-SUB])
finally show cs (to-tf-formula  $\varphi$ )  $\leq$  cs  $\varphi$  .

```



qed

**lemma** *to-tf-mformula*: **assumes**  $\neg \text{eval } \vartheta \ \varphi$   
**shows**  $\exists \psi \in \text{tf-mformula}. (\forall \vartheta. \text{eval } \vartheta \ \varphi = \text{eval } \vartheta \ \psi) \wedge \text{vars } \psi \subseteq \text{vars } \varphi \wedge \text{cs } \psi \leq \text{cs } \varphi$   
**proof** (*intro* *bestI*[*of - to-tf-formula*  $\varphi$ ] *conjI* *allI* *eval-to-tf-formula*[*symmetric*] *vars-to-tf-formula* *to-tf-formula*)  
**from** *assms* **have**  $\neg \text{eval } \vartheta \ (\text{to-tf-formula } \varphi)$  **by** (*simp* *add: eval-to-tf-formula*)  
**thus** *to-tf-formula*  $\varphi \neq \text{TRUE}$  **by** *auto*  
**show**  $\text{cs } (\text{to-tf-formula } \varphi) \leq \text{cs } \varphi$  **by** (*rule cs-to-tf-formula*)  
qed

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**  $l \ p \ k :: \text{nat}$   
**assumes**  $l2: l > 2$   
**and**  $pl: p > l$   
**and**  $kp: k > p$   
**begin**

**lemma**  $k2: k > 2$  **using**  $pl \ l2 \ kp$  **by** *auto*  
**lemma**  $p: p > 2$  **using**  $pl \ l2 \ kp$  **by** *auto*  
**lemma**  $k: k > l$  **using**  $pl \ l2 \ kp$  **by** *auto*

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

**lemma**  $km: k < m$   
**using** *power-strict-increasing-iff*[*of*  $k \ 1 \ 4$ ]  $k2$  **unfolding** *m-def* **by** *auto*

**lemma**  $lm: l + 1 < m$  **using**  $km \ k$  **by** *simp*

**lemma**  $m2: m > 2$  **using**  $k2 \ km$  **by** *auto*

**lemma**  $mp: m > p$  **using**  $km \ k \ kp$  **by** *simp*

**definition**  $L = \text{fact } l * (p - 1) ^ l$

**lemma**  $kml: k \leq m - l$

**proof** –

**have**  $k \leq k * k - k$  **using**  $k2$  **by**  $(\text{cases } k, \text{auto})$

**also have**  $\dots \leq (k * k) * 1 - l$  **using**  $k$  **by**  $\text{simp}$

**also have**  $\dots \leq (k * k) * (k * k) - l$

**by**  $(\text{intro } \text{diff-le-mono } \text{mult-left-mono}, \text{insert } k2, \text{auto})$

**also have**  $(k * k) * (k * k) = m$  **unfolding**  $m\text{-def}$  **by**  $\text{algebra}$

**finally show**  $?thesis$  .

**qed**

**end**

**locale**  $\text{second-assumptions} = \text{first-assumptions} +$

**assumes**  $kl2: k = l^2$

**and**  $l8: l \geq 8$

**begin**

**lemma**  $Lm: L \geq m$

**proof** –

**have**  $m \leq l ^ l$

**unfolding**  $L\text{-def } m\text{-def}$

**unfolding**  $kl2$   $\text{power-mult}[\text{symmetric}]$

**by**  $(\text{intro } \text{power-increasing}, \text{insert } l8, \text{auto})$

**also have**  $\dots \leq (p - 1) ^ l$

**by**  $(\text{rule } \text{power-mono}, \text{insert } pl, \text{auto})$

**also have**  $\dots \leq \text{fact } l * (p - 1) ^ l$  **by**  $\text{simp}$

**also have**  $\dots \leq L$  **unfolding**  $L\text{-def}$  **by**  $\text{simp}$

**finally show**  $?thesis$  .

**qed**

**lemma**  $Lp: L > p$  **using**  $Lm$   $mp$  **by**  $\text{auto}$

**lemma**  $L3: L > 3$  **using**  $p$   $Lp$  **by**  $\text{auto}$

**end**

**definition**  $\text{eps} = 1/(1000 :: \text{real})$

**lemma**  $\text{eps}: \text{eps} > 0$  **unfolding**  $\text{eps-def}$  **by**  $\text{simp}$

**definition**  $L0 :: \text{nat}$  **where**

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

**definition**  $M0 :: \text{nat}$  **where**

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

**definition**  $L0' :: \text{nat}$  **where**

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

$8 * \text{real } (n^2 \wedge 4) \text{ powr } (1 / 8))$

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

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

**have**  $(\lambda l :: \text{nat}. (\text{real } l * \log 2 (\text{real } (l^2 \wedge 4)) + 1) / \text{real } (l^2)) \longrightarrow 0$  **by** *real-asymp*

**from** *LIMSEQ-D*[*OF this, of 1*] **obtain**  $l0$

**where**  $\forall l \geq l0. |1 + \text{real } l * \log 2 (\text{real } (l^2 \wedge 4))| / (\text{real } l)^2 < 1$  **by** (*auto simp: field-simps*)

**hence**  $\forall l \geq \max 1 l0. \text{real } l * \log 2 (\text{real } (l^2 \wedge 4)) + 1 < \text{real } (l^2)$

**by** (*auto simp: field-simps*)

**hence**  $\exists l0. \forall l \geq l0. \text{real } l * \log 2 (\text{real } (l^2 \wedge 4)) + 1 < \text{real } (l^2)$  **by** *blast*

**from** *someI-ex*[*OF this, folded L0''-def, rule-format, OF assms*]

**show** *?thesis* .

**qed**

**definition**  $M0' :: \text{nat}$  **where**

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

**locale** *third-assumptions* = *second-assumptions* +

**assumes** *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** –

**have**  $\text{real } (k - 1) \wedge (m - l) * (\prod i = 0..<l. \text{real } (k - 1 - i)) =$   
 $\text{real } (k - 1) \wedge m *$

$(\text{inverse } (\text{real } (k - 1)) \wedge l * (\prod i = 0..<l. \text{real } (k - 1 - i)))$

**by** (*subst power-diff-conv-inverse, insert k2 lm, auto*)

**also have**  $\dots > (\text{real } (k - 1)) \wedge m * (1/3)$

**proof** (*rule mult-strict-left-mono*)

**define**  $f$  **where**  $f l = (1 + (-1) / \text{real } l) \wedge l$  **for**  $l$

**define**  $e1 :: \text{real}$  **where**  $e1 = \text{exp } (-1)$

**define**  $lim :: \text{real}$  **where**  $lim = 1 / 3$

**from** *tendsto-exp-limit-sequentially*[*of -1, folded f-def*]

**have**  $f: f \longrightarrow e1$  **by** (*simp add: e1-def*)

**have**  $lim < (1 - 1 / \text{real } 6) \wedge 6$  **unfolding** *lim-def* **by** *code-simp*

**also have**  $\dots \leq \text{exp } (-1)$

**by** (*rule exp-ge-one-minus-x-over-n-power-n, auto*)

**finally have**  $lim < e1$  **unfolding** *e1-def* **by** *auto*

```

with f have  $\exists l_0. \forall l. l \geq l_0 \longrightarrow f l > \text{lim}$ 
  by (metis eventually-sequentially order-tendstoD(1))
from someI-ex[OF this[unfolded f-def lim-def], folded L0-def] L0
have fl:  $f l > 1/3$  unfolding f-def by auto
define start where start = inverse (real (k - 1)) ^ l * ( $\prod_{i = 0..<l} \text{real } (k - 1 - i)$ )
  have uminus start
    = uminus (prod ( $\lambda \cdot. \text{inverse } (\text{real } (k - 1))$ ) {0..<l} * prod ( $\lambda i. \text{real } (k - 1 - i)$ ) {0..<l})
    by (simp add: start-def)
  also have ... = uminus (prod ( $\lambda i. \text{inverse } (\text{real } (k - 1)) * \text{real } (k - 1 - i)$ ) {0..<l})
    by (subst prod.distrib, simp)
  also have ...  $\leq$  uminus (prod ( $\lambda i. \text{inverse } (\text{real } (k - 1)) * \text{real } (k - 1 - (l - 1))$ ) {0..<l})
    unfolding neg-le-iff-le
  by (intro prod-mono conjI mult-left-mono, insert k2 l2, auto intro!: diff-le-mono2)
  also have ... = uminus ((inverse (real (k - 1)) * real (k - l)) ^ l) by simp
  also have inverse (real (k - 1)) * real (k - l) = inverse (real (k - 1)) * ((real (k - 1)) - (real l - 1))
    using l2 k2 k by simp
  also have ... = 1 - (real l - 1) / (real (k - 1)) using l2 k2 k
    by (simp add: field-simps)
  also have real (k - 1) = real k - 1 using k2 by simp
  also have ... = (real l - 1) * (real l + 1) unfolding kl2 of-nat-power
    by (simp add: field-simps power2-eq-square)
  also have (real l - 1) / ... = inverse (real l + 1)
    using l2 by (smt (verit, best) divide-divide-eq-left' divide-inverse nat-1-add-1 nat-less-real-le nonzero-mult-div-cancel-left of-nat-1 of-nat-add)
  also have - ((1 - inverse (real l + 1)) ^ l)  $\leq$  - ((1 - inverse (real l)) ^ l)
    unfolding neg-le-iff-le
  by (intro power-mono, insert l2, auto simp: field-simps)
  also have ... < - (1/3) using fl unfolding f-def by (auto simp: field-simps)
  finally have start: start > 1 / 3 by simp
  thus inverse (real (k - 1)) ^ l * ( $\prod_{i = 0..<l} \text{real } (k - 1 - i)$ ) > 1/3
    unfolding start-def by simp
qed (insert k2, auto)
finally show ?thesis by simp
qed

```

```

lemma approximation2: fixes s :: nat
  assumes m choose k  $\leq$  s * L2 * (m - l - 1 choose (k - l - 1))
  shows ((m - l) / k) ^ l / (6 * L2) < s
proof -
  let ?r = real
  define q where q = (?r (L2) * ?r (m - l - 1 choose (k - l - 1)))
  have q: q > 0 unfolding q-def
    by (insert L3 km, auto)
  have ?r (m choose k)  $\leq$  ?r (s * L2 * (m - l - 1 choose (k - l - 1)))

```

**unfolding of-nat-le-iff using assms by simp**  
**hence**  $m$  **choose**  $k \leq s * q$  **unfolding**  $q$ -def **by** *simp*  
**hence**  $*$ :  $s \geq (m \text{ choose } k) / q$  **using**  $q$  **by** (*metis mult-imp-div-pos-le*)  
**have**  $((m - l) / k)^\wedge l / (L^\wedge 2) / 6 < ((m - l) / k)^\wedge l / (L^\wedge 2) / 1$   
**by** (*rule divide-strict-left-mono, insert m2 L3 lm k, auto intro!: mult-pos-pos divide-pos-pos zero-less-power*)  
**also have**  $\dots = ((m - l) / k)^\wedge l / (L^\wedge 2)$  **by** *simp*  
**also have**  $\dots \leq ((m \text{ choose } k) / (m - l - 1 \text{ choose } (k - l - 1))) / (L^\wedge 2)$   
**proof** (*rule divide-right-mono*)  
**define**  $b$  **where**  $b = ?r (m - l - 1 \text{ choose } (k - l - 1))$   
**define**  $c$  **where**  $c = (?r k)^\wedge l$   
**have**  $b0$ :  $b > 0$  **unfolding**  $b$ -def **using**  $km$   $l2$  **by** *simp*  
**have**  $c0$ :  $c > 0$  **unfolding**  $c$ -def **using**  $k$  **by** *auto*  
**define**  $aim$  **where**  $aim = (((m - l) / k)^\wedge l \leq (m \text{ choose } k) / (m - l - 1 \text{ choose } (k - l - 1)))$   
**have**  $aim \iff ((m - l) / k)^\wedge l \leq (m \text{ choose } k) / b$  **unfolding**  $b$ -def  $aim$ -def  
**by** *simp*  
**also have**  $\dots \iff b * ((m - l) / k)^\wedge l \leq (m \text{ choose } k)$  **using**  $b0$   
**by** (*simp add: mult.commute pos-le-divide-eq*)  
**also have**  $\dots \iff b * (m - l)^\wedge l / c \leq (m \text{ choose } k)$   
**by** (*simp add: power-divide c-def*)  
**also have**  $\dots \iff b * (m - l)^\wedge l \leq (m \text{ choose } k) * c$  **using**  $c0$   $b0$   
**by** (*auto simp add: mult.commute pos-divide-le-eq*)  
**also have**  $(m \text{ choose } k) = \text{fact } m / (\text{fact } k * \text{fact } (m - k))$   
**by** (*rule binomial-fact, insert km, auto*)  
**also have**  $b = \text{fact } (m - l - 1) / (\text{fact } (k - l - 1) * \text{fact } (m - l - 1 - (k - l - 1)))$  **unfolding**  $b$ -def  
**by** (*rule binomial-fact, insert k km, auto*)  
**finally have**  $aim \iff$   
 $\text{fact } (m - l - 1) / \text{fact } (k - l - 1) * (m - l)^\wedge l / \text{fact } (m - l - 1 - (k - l - 1))$   
 $\leq (\text{fact } m / \text{fact } k) * (?r k)^\wedge l / \text{fact } (m - k)$  **unfolding**  $c$ -def **by** *simp*  
**also have**  $m - l - 1 - (k - l - 1) = m - k$  **using**  $l2$   $k$   $km$  **by** *simp*  
**finally have**  $aim \iff$   
 $\text{fact } (m - l - 1) / \text{fact } (k - l - 1) * ?r (m - l)^\wedge l$   
 $\leq \text{fact } m / \text{fact } k * ?r k^\wedge l$  **unfolding** *divide-le-cancel* **using**  $km$  **by** *simp*  
**also have**  $\dots \iff (\text{fact } (m - (l + 1)) * ?r (m - l)^\wedge l) * \text{fact } k$   
 $\leq (\text{fact } m / k) * (\text{fact } (k - (l + 1)) * (?r k * ?r k^\wedge l))$   
**using**  $k2$   
**by** (*simp add: field-simps*)  
**also have**  $\dots$   
**proof** (*intro mult-mono*)  
**have**  $\text{fact } k \leq \text{fact } (k - (l + 1)) * (?r k)^\wedge (l + 1)$   
**by** (*rule fact-approx-minus, insert k, auto*)  
**also have**  $\dots = (\text{fact } (k - (l + 1)) * ?r k^\wedge l) * ?r k$  **by** *simp*  
**finally show**  $\text{fact } k \leq \text{fact } (k - (l + 1)) * (?r k * ?r k^\wedge l)$  **by** (*simp add: field-simps*)  
**have**  $\text{fact } (m - (l + 1)) * \text{real } (m - l)^\wedge l \leq \text{fact } m / k \iff$   
 $(\text{fact } (m - (l + 1)) * ?r k) * \text{real } (m - l)^\wedge l \leq \text{fact } m$  **using**  $k2$  **by** (*simp*)

```

add: field-simps)
  also have ...
  proof -
    have (fact (m - (l + 1)) * ?r k) * ?r (m - l) ^ l ≤
      (fact (m - (l + 1)) * ?r (m - l)) * ?r (m - l) ^ l
      by (intro mult-mono, insert kml, auto)
    also have ((fact (m - (l + 1)) * ?r (m - l)) * ?r (m - l) ^ l) =
      (fact (m - (l + 1)) * ?r (m - l) ^ (l + 1)) by simp
    also have ... ≤ fact m
      by (rule fact-approx-upper-minus, insert km k, auto)
    finally show fact (m - (l + 1)) * real k * real (m - l) ^ l ≤ fact m .
  qed
  finally show fact (m - (l + 1)) * real (m - l) ^ l ≤ fact m / k .
  qed auto
  finally show ((m - l) / k) ^ l ≤ (m choose k) / (m - l - 1 choose (k - l -
1))
    unfolding aim-def .
  qed simp
  also have ... = (m choose k) / q
    unfolding q-def by simp
  also have ... ≤ s using q * by metis
  finally show ((m - l) / k) ^ l / (6 * L ^ 2) < s by simp
qed

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 -
    define A where A = real (L^2 * (k - 1) ^ m)
    have A0: A > 0 unfolding A-def using L3 k2 m2 by simp
    from mult-strict-left-mono[OF assms, of 2 ^ (p - 1)]
    have 2 ^ (p - 1) * (k - 1) ^ m / 3 < s * A
      by (simp add: A-def)
    from divide-strict-right-mono[OF this, of A] A0
    have 2 ^ (p - 1) * (k - 1) ^ m / 3 / A < s
      by simp
    also have 2 ^ (p - 1) * (k - 1) ^ m / 3 / A = 2 ^ (p - 1) / (3 * L ^ 2)
      unfolding A-def using k2 by simp
    also have ... = 2 ^ p / (6 * L ^ 2) using p by (cases p, auto)
    also have 2 ^ p = 2 powr p
      by (simp add: powr-realpow)
    finally have *: 2 powr p / (6 * L^2) < s .
    have m ^ l = m powr l using m2 l2 powr-realpow by auto
    also have ... = 2 powr (log 2 m * l)
      unfolding powr-powr[symmetric]
      by (subst powr-log-cancel, insert m2, auto)
    also have ... = 2 powr (l * log 2 m) by (simp add: ac-simps)
    also have ... ≤ 2 powr p
      by (rule powr-mono, insert pllog, auto)
  
```

**finally have**  $m \wedge l \leq 2 \text{ powr } p$  .  
**from** *divide-right-mono*[*OF this, of  $6 * L^2$* ] \*  
**have**  $m \wedge l / (6 * L^2) < s$  **by** *simp*  
**moreover have**  $((m - l) / k) \wedge l / (6 * L^2) \leq m \wedge l / (6 * L^2)$   
**proof** (*rule divide-right-mono, unfold of-nat-power, rule power-mono*)  
  **have**  $\text{real } (m - l) / \text{real } k \leq \text{real } (m - l) / 1$   
  **using** *k2 lm* **by** (*intro divide-left-mono, auto*)  
  **also have**  $\dots \leq m$  **by** *simp*  
  **finally show**  $(m - l) / k \leq m$  **by** *simp*  
**qed** *auto*  
**ultimately show** *?thesis* **by** *simp*  
**qed**

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

**proof** –  
  **let** *?r = real*  
  **have**  $?r \ k \wedge 4 = ?r \ m$  **unfolding** *m-def* **by** *simp*  
  **from** *arg-cong*[*OF this, of root 4*]  
  **show** *km-id*:  $k = \text{root } 4 \ m$  **by** (*simp add: real-root-pos2*)  
  **have**  $?r \ l \wedge 8 = ?r \ m$  **unfolding** *m-def* **using** *kl2* **by** *simp*  
  **from** *arg-cong*[*OF this, of root 8*]  
  **show** *lm-id*:  $l = \text{root } 8 \ m$  **by** (*simp add: real-root-pos2*)  
**qed**

**lemma identities2:**  $\text{root } 4 \ m = m \text{ powr } (1/4) \ \text{root } 8 \ m = m \text{ powr } (1/8)$   
  **by** (*subst root-powr-inverse, insert m2, auto*)**+**

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

**proof** –  
  **have**  $(\lambda x. x \text{ powr } (2/3) / (x \text{ powr } (3/4) - 1)) \longrightarrow 0$   
  **by** *real-asymp*  
  **from** *LIMSEQ-D*[*OF this, of 1, simplified*] **obtain** *x0 :: nat* **where**  
  *sub*:  $x \geq x0 \implies x \text{ powr } (2 / 3) / |x \text{ powr } (3/4) - 1| < 1$  **for** *x*  
  **by** (*auto simp: field-simps*)  
  **have**  $(\lambda x :: \text{real}. 2 / (x \text{ powr } (3/4))) \longrightarrow 0$   
  **by** *real-asymp*  
  **from** *LIMSEQ-D*[*OF this, of 1, simplified*] **obtain** *x1 :: nat* **where**  
  *sub2*:  $x \geq x1 \implies 2 / x \text{ powr } (3 / 4) < 1$  **for** *x* **by** *auto*  
  {  
    **fix** *x*  
    **assume** *x*:  $x \geq x0 \ x \geq x1 \ x \geq 1$   
    **define** *a* **where**  $a = x \text{ powr } (3/4) - 1$   
    **from** *sub*[*OF x(1)*] **have** *small*:  $x \text{ powr } (2 / 3) / |a| \leq 1$   
    **by** (*simp add: a-def*)  
    **have** *2*:  $2 \leq x \text{ powr } (3/4)$  **using** *sub2*[*OF x(2)*] *x(3)* **by** *simp*  
    **hence** *a*:  $a > 0$  **by** (*simp add: a-def*)  
    **from** *mult-left-mono*[*OF small, of a*] *a*

```

    have x powr (2 / 3) ≤ a
      by (simp add: field-simps)
    hence x powr (2 / 3) ≤ x powr (3 / 4) - 1 unfolding a-def by simp
  }
  hence ∃ x0 :: nat. ∀ x ≥ x0. x powr (2 / 3) ≤ x powr (3 / 4) - 1
    by (intro exI[of - max x0 (max x1 1)], auto)
  from someI-ex[OF this, folded M0'-def, rule-format, OF assms]
  show ?thesis .
qed

```

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

**proof** –

```

  define f where f (x :: nat) = (root 8 x * log 2 x + 1) / (x powr (1/8 + eps))
  for x
  have f ⟶ 0 using eps unfolding f-def by real-asymp
  from LIMSEQ-D[OF this, of 1]
  have ex: ∃ x. ∀ y. y ≥ x ⟶ f y ≤ 1 by fastforce
  have lim: root 8 m * log 2 m + 1 ≤ m powr (1 / 8 + eps)
    using someI-ex[OF ex[unfolding f-def], folded M0-def, rule-format, OF M0] m2
    by (simp add: field-simps)
  define start where start = real (p - 1)⌈l⌉
  have (p - 1)⌈l⌉ < p⌈l⌉
    by (rule power-strict-mono, insert p l2, auto)
  hence start < real (p⌈l⌉)
    using start-def of-nat-less-of-nat-power-cancel-iff by blast
  also have ... = p powr l
    by (subst powr-realpow, insert p, auto)
  also have ... ≤ (l * log 2 m + 1) powr l
    by (rule powr-mono2, insert pllog, auto)
  also have l = root 8 m unfolding identities by simp
  finally have start < (root 8 m * log 2 m + 1) powr root 8 m
    by (simp add: identities2)
  also have ... ≤ (m powr (1 / 8 + eps)) powr root 8 m
    by (rule powr-mono2[OF - - lim], insert m2, auto)
  also have ... = m powr ((1 / 8 + eps) * l) unfolding powr-powr identities ..
  finally show ?thesis unfolding start-def by simp
qed

```

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

**proof** –

```

  define f where f = (λ n. 6 * (real n)16 * (sqrt (2 * pi * real n)) * (real n / exp 1)n)
  define g where g = (λ n. 6 * (real n)16 * (sqrt (2 * 4 * real n)) * (real n / 2)n)
  define h where h = (λ n. ((real (n2 ^ 4) powr (1 / 8 * (real (n2 ^ 4)) powr (1/8))))))
  have e: 2 ≤ (exp 1 :: real) using exp-ge-add-one-self[of 1] by simp
  from fact-asymp-equiv

```



**have 1:**  $(\lambda n. 6 * (\text{real } n)^{\wedge}16 * \text{fact } n / h n) \sim[\text{sequentially}] (\lambda n. f n / h n)$   
**unfolding** *f-def*  
**by** (*intro asymp-equiv-intros*)  
**have 2:**  $f n \leq g n$  **for**  $n$  **unfolding** *f-def g-def*  
**by** (*intro mult-mono power-mono divide-left-mono real-sqrt-le-mono, insert pi-less-4 e, auto*)  
**have 2:**  $\text{abs } (f n / h n) \leq \text{abs } (g n / h n)$  **for**  $n$   
**unfolding** *abs-le-square-iff power2-eq-square*  
**by** (*intro mult-mono divide-right-mono 2, auto simp: h-def f-def g-def*)  
**have 2:**  $\text{abs } (g n / h n) < e \implies \text{abs } (f n / h n) < e$  **for**  $n e$  **using**  $2[\text{of } n]$  **by**  
*simp*  
**have**  $(\lambda n. g n / h n) \longrightarrow 0$   
**unfolding** *g-def h-def* **by** *real-asymp*  
**from** *LIMSEQ-D[OF this] 2*  
**have**  $(\lambda n. f n / h n) \longrightarrow 0$   
**by** (*intro LIMSEQ-I, fastforce*)  
**with 1** **have**  $(\lambda n. 6 * (\text{real } n)^{\wedge}16 * \text{fact } n / h n) \longrightarrow 0$   
**using** *tendsto-asymp-equiv-cong* **by** *blast*  
**from** *LIMSEQ-D[OF this, of 1]* **obtain**  $n0$  **where**  $3: n \geq n0 \implies \text{norm } (6 * (\text{real } n)^{\wedge}16 * \text{fact } n / h n) < 1$  **for**  $n$  **by** *auto*  
{  
**fix**  $n$   
**assume**  $n: n \geq \text{max } 1 n0$   
**hence**  $hn: h n > 0$  **unfolding** *h-def* **by** *auto*  
**from**  $n$  **have**  $n \geq n0$  **by** *simp*  
**from**  $3[\text{OF this}]$  **have**  $6 * n^{\wedge}16 * \text{fact } n / \text{abs } (h n) < 1$  **by** *auto*  
**with**  $hn$  **have**  $6 * (\text{real } n)^{\wedge}16 * \text{fact } n < h n$  **by** *simp*  
}  
**hence**  $\exists n0. \forall n. n \geq n0 \longrightarrow 6 * n^{\wedge}16 * \text{fact } n < h n$  **by** *blast*  
**from** *someI-ex[OF this[unfolded h-def], folded L0'-def, rule-format, OF L0']*  
**have**  $6 * \text{real } l^{\wedge}16 * \text{fact } l < \text{real } (l^2)^{\wedge}4 \text{ powr } (1 / 8 * \text{real } (l^2)^{\wedge}4) \text{ powr } (1 / 8)$   
**by** *simp*  
**also** **have**  $\dots = m \text{ powr } (1 / 8 * l)$  **using** *identities identities2 kl2*  
**by** (*metis m-def*)  
**finally** **show** *?thesis* .  
**qed**

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

**proof** –

**let**  $?r = \text{real}$

**define** *Lappr* **where**  $Lappr = m * m * \text{fact } l * p^{\wedge}l / 2$

**have**  $L = (\text{fact } l * (p - 1)^{\wedge}l)$  **unfolding** *L-def* **by** *simp*

**hence**  $?r L \leq (\text{fact } l * (p - 1)^{\wedge}l)$  **by** *linarith*

**also** **have**  $\dots = (1 * ?r (\text{fact } l)) * (?r (p - 1)^{\wedge}l)$  **by** *simp*

**also** **have**  $\dots \leq ((m * m / 2) * ?r (\text{fact } l)) * (?r (p - 1)^{\wedge}l)$

**by** (*intro mult-right-mono, insert m2, cases m; cases m - 1, auto*)

**also** **have**  $\dots = (6 * \text{real } (m * m) * \text{fact } l) * (?r (p - 1)^{\wedge}l) / 12$  **by** *simp*

**also** **have**  $\text{real } (m * m) = \text{real } l^{\wedge}16$  **unfolding** *m-def* **unfolding** *kl2* **by** *simp*

**also** **have**  $(6 * \text{real } l^{\wedge}16 * \text{fact } l) * (?r (p - 1)^{\wedge}l) / 12$

$\leq (m \text{ powr } (1 / 8 * l) * (m \text{ powr } ((1 / 8 + \text{eps}) * l))) / 12$   
**by** (*intro divide-right-mono mult-mono, insert appendix-A-2 appendix-A-3, auto*)

**also have**  $\dots = (m \text{ powr } (1 / 8 * l + (1 / 8 + \text{eps}) * l)) / 12$   
**by** (*simp add: powr-add*)

**also have**  $1 / 8 * l + (1 / 8 + \text{eps}) * l = l * (1/4 + \text{eps})$  **by** (*simp add: field-simps*)

**also have**  $l = m \text{ powr } (1/8)$  **unfolding identities identities2 ..**

**finally have**  $LL: ?r L \leq m \text{ powr } (m \text{ powr } (1 / 8) * (1 / 4 + \text{eps})) / 12 .$

**from** *power-mono[OF this, of 2]*

**have**  $L^{\wedge}2 \leq (m \text{ powr } (m \text{ powr } (1 / 8) * (1 / 4 + \text{eps})) / 12)^{\wedge}2$   
**by** *simp*

**also have**  $\dots = (m \text{ powr } (m \text{ powr } (1 / 8) * (1 / 4 + \text{eps})))^{\wedge}2 / 144$   
**by** (*simp add: power2-eq-square*)

**also have**  $\dots = (m \text{ powr } (m \text{ powr } (1 / 8) * (1 / 4 + \text{eps}) * 2)) / 144$   
**by** (*subst powr-realpow[symmetric], (use m2 in force), unfold powr-powr, simp*)

**also have**  $\dots = (m \text{ powr } (m \text{ powr } (1 / 8) * (1 / 2 + 2 * \text{eps}))) / 144$   
**by** (*simp add: algebra-simps*)

**also have**  $\dots \leq (m \text{ powr } (m \text{ powr } (1 / 8) * 0.51)) / 144$   
**by** (*intro divide-right-mono powr-mono mult-left-mono, insert m2, auto simp: eps-def*)

**finally have**  $L^{\wedge}2 \leq m \text{ powr } (m \text{ powr } (1 / 8) * 0.51) / 144$  **by** *simp*

**from** *mult-left-mono[OF this, of 12]*

**have**  $12 * L^{\wedge}2 \leq 12 * m \text{ powr } (m \text{ powr } (1 / 8) * 0.51) / 144$  **by** *simp*

**also have**  $\dots = m \text{ powr } (m \text{ powr } (1 / 8) * 0.51) / 12$  **by** *simp*

**also have**  $\dots \leq m \text{ powr } (m \text{ powr } (1 / 8) * 0.51) / 1$   
**by** (*rule divide-left-mono, auto*)

**finally show** *?thesis* **by** *simp*

**qed**

**lemma approximation4: fixes**  $s :: \text{nat}$

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

**shows**  $s > 2 * k \text{ powr } (4 / 7 * \text{sqrt } k)$

**proof** –

**let**  $?r = \text{real}$

**have** *diff: ?r (m - l) = ?r m - ?r l* **using** *lm* **by** *simp*

**have**  $m \text{ powr } (2/3) \leq m \text{ powr } (3/4) - 1$  **using** *appendix-A-1[OF M0]* **by** *auto*

**also have**  $\dots \leq (m - m \text{ powr } (1/8)) / m \text{ powr } (1/4)$   
**unfolding** *diff-divide-distrib*

**by** (*rule diff-mono, insert m2, auto simp: divide-powr-uminus powr-mult-base powr-add[symmetric], auto simp: powr-minus-divide intro!: ge-one-powr-ge-zero*)

**also have**  $\dots = (m - \text{root } 8 m) / \text{root } 4 m$  **using** *m2*

**by** (*simp add: root-powr-inverse*)

**also have**  $\dots = (m - l) / k$  **unfolding identities diff** **by** *simp*

**finally have**  $m \text{ powr } (2/3) \leq (m - l) / k$  **by** *simp*

**from** *power-mono[OF this, of l]*

**have** *ineq1: (m powr (2 / 3))^{\wedge}l \leq ((m - l) / k)^{\wedge}l* **using** *m2* **by** *auto*

**have**  $(m \text{ powr } (l / 7)) \leq (m \text{ powr } (2 / 3 * l - l * 0.51))$

```

    by (intro powr-mono, insert m2, auto)
  also have ... = (m powr (2 / 3)) powr l / (m powr (m powr (1 / 8) * 0.51))
    unfolding powr-diff powr-powr identities identities2 by simp
  also have ... = (m powr (2 / 3)) ^ l / (m powr (m powr (1 / 8) * 0.51))
    by (subst powr-realpow, insert m2, auto)
  also have ... ≤ (m powr (2 / 3)) ^ l / (12 * L2)
    by (rule divide-left-mono[OF appendix-A-4], insert L3 m2, auto intro!: mult-pos-pos)
  also have ... = (m powr (2 / 3)) ^ l / (?r 12 * L2) by simp
  also have ... ≤ ((m - l) / k) ^ l / (?r 12 * L2)
    by (rule divide-right-mono[OF ineq1], insert L3, auto)
  also have ... < s / 2 using assms by simp
  finally have 2 * m powr (real l / 7) < s by simp
  also have m powr (real l / 7) = m powr (root 8 m / 7)
    unfolding identities by simp
  finally have s > 2 * m powr (root 8 m / 7) by simp
  also have root 8 m = root 2 k using m2
    by (metis identities(2) kl2 of-nat-0-le-iff of-nat-power pos2 real-root-power-cancel)
  also have ?r m = k powr 4 unfolding m-def by simp
  also have (k powr 4) powr ((root 2 k) / 7)
    = k powr (4 * (root 2 k) / 7) unfolding powr-powr by simp
  also have ... = k powr (4 / 7 * sqrt k) unfolding sqrt-def by simp
  finally show s > 2 * k powr (4 / 7 * sqrt k) .
qed

```

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

$$X \cdot Y = \{\{x, y\} \mid x \in X \wedge y \in Y \wedge x \neq y\}$$

abbreviation *sameprod* :: 'a set ⇒ 'a set set ((-) ^ 2) where

$$X^2 \equiv X \cdot X$$

**lemma sameprod-altdef:**  $X^{\mathbf{2}} = \{Y. Y \subseteq X \wedge \text{card } Y = 2\}$   
**unfolding** *binprod-def* **by** (*auto simp: card-2-iff*)

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

**lemma card-sameprod:**  $\text{finite } X \Longrightarrow \text{card } (X^{\mathbf{2}}) = \text{card } X \text{ choose } 2$   
**unfolding** *sameprod-altdef*  
**by** (*subst n-subsets, auto*)

**lemma sameprod-mono:**  $X \subseteq Y \Longrightarrow X^{\mathbf{2}} \subseteq Y^{\mathbf{2}}$   
**unfolding** *sameprod-altdef* **by** *auto*

**lemma sameprod-finite:**  $\text{finite } X \Longrightarrow \text{finite } (X^{\mathbf{2}})$   
**unfolding** *sameprod-altdef* **by** *simp*

**lemma numbers2-mono:**  $x \leq y \Longrightarrow [x]^{\mathbf{2}} \subseteq [y]^{\mathbf{2}}$   
**by** (*rule sameprod-mono, auto simp: numbers-def*)

**lemma card-numbers[*simp*]:**  $\text{card } [n] = n$   
**by** (*simp add: numbers-def*)

**lemma card-numbers2[*simp*]:**  $\text{card } ([n]^{\mathbf{2}}) = n \text{ choose } 2$   
**by** (*subst card-sameprod, auto simp: numbers-def*)

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

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

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

**context** *first-assumptions*  
**begin**

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

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

**lemma empty- $\mathcal{G}$ [*simp*]:**  $\{\} \in \mathcal{G}$  **unfolding**  $\mathcal{G}\text{-def}$  **by** *auto*

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

**lemma v-union:**  $v (G \cup H) = v G \cup v H$   
**unfolding**  $v\text{-def}$  **by** *auto*

**definition**  $\mathcal{K}$  :: *graph set* **where**

$$\mathcal{K} = \{ K . K \in \mathcal{G} \wedge \text{card } (v K) = k \wedge K = (v K) \hat{\ }^2 \}$$

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

**unfolding**  $v\text{-def}$   $\mathcal{G}\text{-def}$   $\text{sameprod}\text{-altdef}$  **by** *auto*

**lemma**  $v\text{-mono}$ :  $G \subseteq H \implies v G \subseteq v H$  **unfolding**  $v\text{-def}$  **by** *auto*

**lemma**  $v\text{-sameprod}$ [*simp*]: **assumes**  $\text{card } X \geq 2$

**shows**  $v (X \hat{\ }^2) = X$

**proof** –

**from** *obtain-subset-with-card-n*[*OF* *assms*] **obtain**  $Y$  **where**  $Y \subseteq X$

**and**  $Y$ :  $\text{card } Y = 2$  **by** *auto*

**then obtain**  $x y$  **where**  $x \in X$   $y \in X$  **and**  $x \neq y$

**by** (*auto simp: card-2-iff*)

**thus** *?thesis* **unfolding**  $\text{sameprod}\text{-altdef}$   $v\text{-def}$

**by** (*auto simp: card-2-iff doubleton-eq-iff*) *blast*

**qed**

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

**proof** –

**obtain**  $x y$  **where**  $e = \{x, y\}$  **and**  $xy$ :  $x \neq y$  **using** *assms*

**by** (*auto simp: card-2-iff*)

**from** *assms*(2) **have**  $x$ :  $x \in v G$  **unfolding**  $e$

**by** (*auto simp: v-def*)

**from**  $e$  **have**  $e = \{y, x\}$  **unfolding**  $e$  **by** *auto*

**from** *assms*(2) **have**  $y$ :  $y \in v G$  **unfolding**  $e$

**by** (*auto simp: v-def*)

**show**  $e \subseteq v G$  **using**  $x y$  **unfolding**  $e$  **by** *auto*

**qed**

**lemma**  $v\text{-}\mathcal{G}\text{-}2$ : **assumes**  $G \in \mathcal{G}$  **shows**  $G \subseteq (v G) \hat{\ }^2$

**proof**

**fix**  $e$

**assume**  $eG$ :  $e \in G$

**with** *assms*[*unfolded*  $\mathcal{G}\text{-def}$  *binprod-def*] **obtain**  $x y$  **where**  $e = \{x, y\}$  **and**  $xy$ :  
 $x \neq y$  **by** *auto*

**from**  $eG$   $e xy$  **have**  $x$ :  $x \in v G$  **by** (*auto simp: v-def*)

**from**  $e$  **have**  $e = \{y, x\}$  **unfolding**  $e$  **by** *auto*

**from**  $eG$   $e xy$  **have**  $y$ :  $y \in v G$  **by** (*auto simp: v-def*)

**from**  $x y xy$  **show**  $e \in (v G) \hat{\ }^2$  **unfolding** *binprod-def*  $e$  **by** *auto*

**qed**

**lemma**  $v\text{-numbers}2$ [*simp*]:  $x \geq 2 \implies v ([x] \hat{\ }^2) = [x]$

**by** (*rule v-sameprod, auto*)

**lemma**  $\text{sameprod}\text{-}\mathcal{G}$ : **assumes**  $X \subseteq [m]$   $\text{card } X \geq 2$

**shows**  $X^{\mathbf{2}} \in \mathcal{G}$   
**unfolding**  $\mathcal{G}$ -def **using**  $assms(2)$   $sameprod-mono[OF\ assms(1)]$   
**by** *auto*

**lemma**  $finite-numbers[simp,intro]: finite\ [n]$   
**unfolding**  $numbers-def$  **by** *auto*

**lemma**  $finite-numbers2[simp,intro]: finite\ ([n]^{\mathbf{2}})$   
**unfolding**  $sameprod-altdef$  **using**  $finite-subset[of\ -\ [m]]$  **by** *auto*

**lemma**  $finite-members-\mathcal{G}: G \in \mathcal{G} \implies finite\ G$   
**unfolding**  $\mathcal{G}$ -def **using**  $finite-subset[of\ G\ [m]^{\mathbf{2}}]$  **by** *auto*

**lemma**  $finite-\mathcal{G}[simp,intro]: finite\ \mathcal{G}$   
**unfolding**  $\mathcal{G}$ -def **by** *simp*

**lemma**  $finite-vG: assumes\ G \in \mathcal{G}$   
**shows**  $finite\ (v\ G)$   
**proof** –  
**from**  $finite-members-\mathcal{G}[OF\ assms]$   
**show** *?thesis*  
**proof** (*induct rule: finite-induct*)  
**case** ( $insert\ xy\ F$ )  
**show** *?case*  
**proof** ( $cases\ \exists\ x\ y.\ xy = \{x,y\}$ )  
**case** *False*  
**hence**  $v\ (insert\ xy\ F) = v\ F$  **unfolding**  $v-def$  **by** *auto*  
**thus** *?thesis* **using**  $insert$  **by** *auto*  
**next**  
**case** *True*  
**then obtain**  $x\ y$  **where**  $xy: xy = \{x,y\}$  **by** *auto*  
**hence**  $v\ (insert\ xy\ F) = insert\ x\ (insert\ y\ (v\ F))$   
**unfolding**  $v-def$  **by** *auto*  
**thus** *?thesis* **using**  $insert$  **by** *auto*  
**qed**  
**qed** (*auto simp: v-def*)  
**qed**

**lemma**  $v-empty[simp]: v\ \{\} = \{\}$  **unfolding**  $v-def$  **by** *auto*

**lemma**  $v-card2: assumes\ G \in \mathcal{G}\ G \neq \{\}$   
**shows**  $2 \leq card\ (v\ G)$   
**proof** –  
**from**  $assms[unfolded\ \mathcal{G}-def]$  **obtain**  $edge$  **where**  $*: edge \in G\ edge \in [m]^{\mathbf{2}}$  **by** *auto*  
**then obtain**  $x\ y$  **where**  $edge: edge = \{x,y\}\ x \neq y$  **unfolding**  $binprod-def$  **by** *auto*  
**with**  $*$  **have**  $sub: \{x,y\} \subseteq v\ G$  **unfolding**  $v-def$   
**by** (*smt (verit, best) insert-commute insert-compr mem-Collect-eq singleton-iff*)

subsetI)  
**from** *assms* *finite-vG* **have** *finite* (v G) **by** *auto*  
**from** *sub*  $\langle x \neq y \rangle$  **this show**  $2 \leq \text{card } (v G)$   
**by** (*metis card-2-iff card-mono*)  
**qed**

**lemma**  $\mathcal{K}$ -altdef:  $\mathcal{K} = \{V^{\wedge}2 \mid V. V \subseteq [m] \wedge \text{card } V = k\}$   
(is - = ?R)  
**proof** -  
{  
**fix** K  
**assume**  $K \in \mathcal{K}$   
**hence**  $K: K \in \mathcal{G}$  **and** *card*:  $\text{card } (v K) = k$  **and**  $KvK: K = (v K)^{\wedge}2$   
**unfolding**  $\mathcal{K}$ -def **by** *auto*  
**from**  $v\mathcal{G}[OF K]$  *card*  $KvK$  **have**  $K \in ?R$  **by** *auto*  
}  
**moreover**  
{  
**fix** V  
**assume**  $1: V \subseteq [m]$  **and**  $\text{card } V = k$   
**hence**  $V^{\wedge}2 \in \mathcal{K}$  **unfolding**  $\mathcal{K}$ -def **using**  $k2$  *sameprod-G[OF 1]*  
**by** *auto*  
}  
**ultimately show** *?thesis* **by** *auto*  
**qed**

**lemma**  $\mathcal{K}$ -G:  $\mathcal{K} \subseteq \mathcal{G}$   
**unfolding**  $\mathcal{K}$ -def **by** *auto*

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

**lemma** *empty-CLIQUE[simp]*:  $\{\} \notin CLIQUE$  **unfolding** *CLIQUE-def*  $\mathcal{K}$ -def **using**  $k2$  **by** (*auto simp: v-def*)

### 4.3 Test Graphs

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

**abbreviation**  $POS \equiv \mathcal{K}$

**lemma**  $POS\mathcal{G}$ :  $POS \subseteq \mathcal{G}$  **by** (*rule*  $\mathcal{K}\mathcal{G}$ )

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 F*  
**unfolding** *F-def numbers-def*  
**by** (*meson finite-PiE finite-lessThan*)

**definition** *C* :: *colorf*  $\Rightarrow$  *graph* **where**  
 $C\ f = \{ \{x, y\} \mid x\ y \cdot \{x, y\} \in [m]^{\wedge 2} \wedge f\ x \neq f\ y \}$

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

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

**proof** –

```

{
  fix G
  assume GC: G ∈ CLIQUE and GN: G ∈ NEG
  from GC[unfolding CLIQUE-def] obtain K where
    K: K ∈ K and G: G ∈ G and KsubG: K ⊆ G by auto
  from GN[unfolding NEG-def] obtain f where fF: f ∈ F and
    GCf: G = C f by auto
  from K[unfolding K-def] have KG: K ∈ G and
    KvK: K = v K2 and card1: card (v K) = k by auto
  from k2 card1 have ineq: card (v K) > card [k - 1] by auto
  from vG[OF KG] have vKm: v K ⊆ [m] by auto
  from fF[unfolding F-def] vKm have f: f ∈ v K → [k - 1]
    by auto
  from card-inj[OF f] ineq
  have ¬ inj-on f (v K) by auto
  then obtain x y where *: x ∈ v K y ∈ v K x ≠ y and ineq: f x = f y
    unfolding inj-on-def by auto
  have {x,y} ∉ G unfolding GCf C-def using ineq
    by (auto simp: doubleton-eq-iff)
  with KsubG KvK have {x,y} ∉ v K2 by auto
  with * have False unfolding binprod-def by auto
}
thus ?thesis by auto
qed

```

**lemma** *NEG-G*:  $NEG \subseteq G$

**proof** –

```

{
  fix f
  assume f ∈ F
  hence Cf ∈ G
    unfolding NEG-def C-def G-def
    by (auto simp: sameprod-altdef)
}
thus NEG ⊆ G unfolding NEG-def by auto

```



qed

**lemma** *finite-POS-NEG*:  $finite (POS \cup NEG)$   
using *POS-G NEG-G*  
by (*intro finite-subset[OF - finite-G], auto*)

**lemma** *POS-sub-CLIQUE*:  $POS \subseteq CLIQUE$   
unfolding *CLIQUE-def* using *K-G* by *auto*

**lemma** *POS-CLIQUE*:  $POS \subset CLIQUE$

**proof** –

have  $[k+1] \sim \mathbf{2} \in CLIQUE$

unfolding *CLIQUE-def*

**proof** (*standard, intro conjI bexI[of - [k]  $\sim \mathbf{2}$ ]*)

show  $[k] \sim \mathbf{2} \subseteq [k+1] \sim \mathbf{2}$

by (*rule numbers2-mono, auto*)

show  $[k] \sim \mathbf{2} \in \mathcal{K}$  unfolding *K-altdef* using *km*

by (*auto intro!: exI[of - [k]], auto simp: numbers-def*)

show  $[k+1] \sim \mathbf{2} \in \mathcal{G}$  using *km k2*

by (*intro sameprod-G, auto simp: numbers-def*)

qed

moreover have  $[k+1] \sim \mathbf{2} \notin POS$  unfolding *K-def* using *v-numbers2[of k + 1]*  
*k2*

by *auto*

ultimately show *?thesis* using *POS-sub-CLIQUE* by *blast*

qed

**lemma** *card-POS*:  $card POS = m$  choose *k*

**proof** –

have  $m$  choose  $k =$

$card \{B. B \subseteq [m] \wedge card B = k\}$  (*is - = card ?A*)

by (*subst n-subsets[of [m] k], auto simp: numbers-def*)

also have  $\dots = card (sameprod ' ?A)$

**proof** (*rule card-image[symmetric]*)

{

fix *A*

assume  $A \in ?A$

hence  $v (sameprod A) = A$  using *k2*

by (*subst v-sameprod, auto*)

}

thus *inj-on* *sameprod ?A* by (*rule inj-on-inverseI*)

qed

also have  $sameprod ' \{B. B \subseteq [m] \wedge card B = k\} = POS$

unfolding *K-altdef* by *auto*

finally show *?thesis* by *simp*

qed

#### 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]*:  $G \in \mathcal{G} \Longrightarrow H \in \mathcal{G} \Longrightarrow G \cup H \in \mathcal{G}$   
**unfolding** *G-def* **by** *auto*

**lemma** *odot-G*:  $X \subseteq \mathcal{G} \Longrightarrow Y \subseteq \mathcal{G} \Longrightarrow X \odot Y \subseteq \mathcal{G}$   
**unfolding** *odot-def* **by** *auto*

#### 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]*:  $D \subseteq G \Longrightarrow D \in X \Longrightarrow X \Vdash G$   
**unfolding** *accepts-def* **by** *auto*

**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*:  $ACC\text{-}cf\ X \subseteq \mathcal{F}$   
**unfolding** *ACC-cf-def* **by** *auto*

**lemma** *finite-ACC[intro,simp]*: *finite* ( $ACC\text{-}cf\ X$ )  
**by** (*rule* *finite-subset[OF ACC-cf-F finite-F]*)

**lemma** *ACC-I[intro]*:  $G \in \mathcal{G} \Longrightarrow X \Vdash G \Longrightarrow G \in ACC\ X$   
**unfolding** *ACC-def* **by** *auto*

**lemma** *ACC-cf-I[intro]*:  $F \in \mathcal{F} \Longrightarrow X \Vdash C\ F \Longrightarrow F \in ACC\text{-}cf\ X$   
**unfolding** *ACC-cf-def* **by** *auto*

**lemma** *ACC-cf-mono*:  $X \subseteq Y \Longrightarrow ACC\text{-}cf\ X \subseteq ACC\text{-}cf\ Y$   
**unfolding** *ACC-cf-def* *accepts-def* **by** *auto*

Lemma 3

**lemma** *ACC-cf-empty*:  $ACC\text{-}cf\ \{\} = \{\}$   
**unfolding** *ACC-cf-def* *accepts-def* **by** *auto*

**lemma** *ACC-empty[simp]*:  $ACC\ \{\} = \{\}$   
**unfolding** *ACC-def* *accepts-def* **by** *auto*

**lemma** *ACC-cf-union*:  $ACC\text{-}cf (X \cup Y) = ACC\text{-}cf X \cup ACC\text{-}cf Y$   
**unfolding** *ACC-cf-def accepts-def* **by** *blast*

**lemma** *ACC-union*:  $ACC (X \cup Y) = ACC X \cup ACC Y$   
**unfolding** *ACC-def accepts-def* **by** *blast*

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

**proof** –

```

{
  fix G
  assume G ∈ ACC (X ⊙ Y)
  from this[unfolded ACC-def accepts-def]
  obtain D E F :: graph where *: D ∈ X E ∈ Y G ∈ G D ∪ E ⊆ G
    by (force simp: odot-def)
  hence G ∈ ACC X ∩ ACC Y
    unfolding ACC-def accepts-def by auto
}
moreover
{
  fix G
  assume G ∈ ACC X ∩ ACC Y
  from this[unfolded ACC-def accepts-def]
  obtain D E where *: D ∈ X E ∈ Y G ∈ G D ⊆ G E ⊆ G
    by auto
  let ?F = D ∪ E
  from * have ?F ∈ X ⊙ Y unfolding odot-def using * by blast
  moreover have ?F ⊆ G using * by auto
  ultimately have G ∈ ACC (X ⊙ Y) using *
    unfolding ACC-def accepts-def by blast
}
ultimately show ?thesis by blast
qed

```

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

**proof** –

```

{
  fix G
  assume G ∈ ACC-cf (X ⊙ Y)
  from this[unfolded ACC-cf-def accepts-def]
  obtain D E :: graph where *: D ∈ X E ∈ Y G ∈ F D ∪ E ⊆ C G
    by (force simp: odot-def)
  hence G ∈ ACC-cf X ∩ ACC-cf Y
    unfolding ACC-cf-def accepts-def by auto
}
moreover
{
  fix F
  assume F ∈ ACC-cf X ∩ ACC-cf Y
  from this[unfolded ACC-cf-def accepts-def]

```

**obtain**  $D E$  **where**  $*$ :  $D \in X E \in Y F \in \mathcal{F} D \subseteq C F E \subseteq C F$   
**by** *auto*  
**let**  $?F = D \cup E$   
**from**  $*$  **have**  $?F \in X \odot Y$  **unfolding** *odot-def* **using**  $*$  **by** *blast*  
**moreover** **have**  $?F \subseteq C F$  **using**  $*$  **by** *auto*  
**ultimately** **have**  $F \in ACC\text{-}cf (X \odot Y)$  **using**  $*$   
**unfolding** *ACC-cf-def accepts-def* **by** *blast*  
**}**  
**ultimately** **show** *?thesis* **by** *blast*  
**qed**

## 4.6 Approximations and deviations

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

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

**lemma** *v-gs-empty[simp]*:  $v\text{-}gs \{ \} = \{ \}$   
**unfolding** *v-gs-def* **by** *auto*

**lemma** *v-gs-union*:  $v\text{-}gs (X \cup Y) = v\text{-}gs X \cup v\text{-}gs Y$   
**unfolding** *v-gs-def* **by** *auto*

**lemma** *v-gs-mono*:  $X \subseteq Y \Longrightarrow v\text{-}gs X \subseteq v\text{-}gs Y$   
**using** *v-gs-def* **by** *auto*

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

**proof** –  
**have**  $v\text{-}gs X \subseteq v \text{ ' } \mathcal{G}$   
**using** *assms* **unfolding** *v-gs-def* **by** *force*  
**moreover** **have** *finite*  $\mathcal{G}$  **using** *finite- $\mathcal{G}$*  **by** *auto*  
**ultimately** **show** *?thesis* **by** (*metis finite-surj*)  
**qed**

**lemma** *finite-v-gs-Gl*: **assumes**  $X \subseteq \mathcal{G}l$   
**shows** *finite* ( $v\text{-}gs X$ )  
**by** (*rule finite-v-gs, insert assms, auto simp: Gl-def*)

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

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

**lemma** *joinl-join*:  $X \odot l Y \subseteq X \odot Y$   
**unfolding** *odot-def odotl-def* **by** *blast*

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

**proof** –

**note**  $\text{fin} = \text{finite-v-gs}[OF X] \text{ finite-v-gs}[OF Y]$   
**have**  $\text{card } (v\text{-gs } Z) \leq \text{card } ((\lambda (A, B). A \cup B) \text{ ` } (v\text{-gs } X \times v\text{-gs } Y))$   
**proof** (*rule card-mono[OF finite-imageI]*)  
**show**  $\text{finite } (v\text{-gs } X \times v\text{-gs } Y)$   
**using**  $\text{fin}$  **by** *auto*  
**have**  $v\text{-gs } Z \subseteq v\text{-gs } (X \odot Y)$   
**using** *v-gs-mono[OF Z]* .  
**also have**  $\dots \subseteq (\lambda(x, y). x \cup y) \text{ ` } (v\text{-gs } X \times v\text{-gs } Y)$  (**is**  $?L \subseteq ?R$ )  
**unfolding** *odot-def v-gs-def* **by** (*force split: if-splits simp: v-union*)  
**finally show**  $v\text{-gs } Z \subseteq (\lambda(x, y). x \cup y) \text{ ` } (v\text{-gs } X \times v\text{-gs } Y)$  .  
**qed**  
**also have**  $\dots \leq \text{card } (v\text{-gs } X \times v\text{-gs } Y)$   
**by** (*rule card-image-le, insert fin, auto*)  
**also have**  $\dots = \text{card } (v\text{-gs } X) * \text{card } (v\text{-gs } Y)$   
**by** (*rule card-cartesian-product*)  
**finally show** *?thesis* .

**qed**

Definition 6 – elementary plucking step

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

**context** *second-assumptions*  
**begin**

Lemma 9 – for elementary plucking step

**lemma** *v-sameprod-subset*:  $v (Vs \hat{\ } \mathbf{2}) \subseteq Vs$  **unfolding** *binprod-def v-def*  
**by** (*auto simp: doubleton-eq-iff*)

**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**  $\text{card } (v\text{-gs } Y) \leq \text{card } (v\text{-gs } X) - p + 1$   
 $Y \subseteq \mathcal{G}l$   
 $POS \cap ACC X \subseteq ACC Y$   
 $2 \hat{\ } p * \text{card } (ACC\text{-cf } Y - ACC\text{-cf } X) \leq (k - 1) \hat{\ } m$

$Y \neq \{\}$   
**proof** –  
**let**  $?vXp = v\text{-gs } X$   
**have**  $sf\text{-precond}: \forall A \in ?vXp. \text{finite } A \wedge \text{card } A \leq l$   
**using**  $X$  **unfolding**  $Gl\text{-def } Gl\text{-def } v\text{-gs}\text{-def}$  **by**  $(\text{auto intro: finite}\text{-}v\mathcal{G} \text{ intro!: } v\mathcal{G} \text{ v}\text{-card2})$   
**note**  $\text{sunflower} = \text{Erdos-Rado-sunflower}[OF \text{ sf}\text{-precond}]$   
**from**  $p$  **have**  $p0: p \neq 0$  **by**  $\text{auto}$   
**have**  $(p - 1) \wedge l * \text{fact } l < \text{card } ?vXp$  **using**  $L[\text{unfolded } L\text{-def}]$   
**by**  $(\text{simp add: ac}\text{-simps})$   
**note**  $\text{sunflower} = \text{sunflower}[OF \text{ this}]$   
**define**  $S$  **where**  $S = (\text{SOME } S. S \subseteq ?vXp \wedge \text{sunflower } S \wedge \text{card } S = p)$   
**define**  $U$  **where**  $U = \{E \in X. v E \in S\}$   
**define**  $Vs$  **where**  $Vs = \bigcap S$   
**define**  $Gs$  **where**  $Gs = Vs \wedge 2$   
**let**  $?U = U$   
**let**  $?New = Gs :: \text{graph}$   
**have**  $Y: Y = X - U \cup \{?New\}$   
**using**  $Y[\text{unfolded plucking}\text{-step}\text{-def } Let\text{-def}, \text{folded } S\text{-def}, \text{folded } U\text{-def}, \text{folded } Vs\text{-def}, \text{folded } Gs\text{-def}]$  .  
**have**  $U: U \subseteq \mathcal{G}l$  **using**  $X$  **unfolding**  $U\text{-def}$  **by**  $\text{auto}$   
**hence**  $U \subseteq \mathcal{G}$  **unfolding**  $Gl\text{-def}$  **by**  $\text{auto}$   
**from**  $\text{sunflower}$   
**have**  $\exists S. S \subseteq ?vXp \wedge \text{sunflower } S \wedge \text{card } S = p$  **by**  $\text{auto}$   
**from**  $\text{someI}\text{-ex}[OF \text{ this}, \text{folded } S\text{-def}]$   
**have**  $S: S \subseteq ?vXp \text{ sunflower } S \text{ card } S = p$  **by**  $(\text{auto simp: } Vs\text{-def})$   
**have**  $\text{fin1}: \text{finite } ?vXp$  **using**  $\text{finite}\text{-}v\text{-gs}\text{-}Gl[OF X]$  .  
**from**  $X$  **have**  $\text{finX}: \text{finite } X$  **unfolding**  $Gl\text{-def}$   
**using**  $\text{finite}\text{-subset}[of X, OF - \text{finite}\mathcal{G}]$  **by**  $\text{auto}$   
**from**  $\text{fin1 } S$  **have**  $\text{finS}: \text{finite } S$  **by**  $(\text{metis finite}\text{-subset})$   
**from**  $\text{finite}\text{-subset}[OF - \text{finX}]$  **have**  $\text{finU}: \text{finite } U$  **unfolding**  $U\text{-def}$  **by**  $\text{auto}$   
**from**  $S p$  **have**  $S\text{empty}: S \neq \{\}$  **by**  $\text{auto}$   
**have**  $UX: U \subseteq X$  **unfolding**  $U\text{-def}$  **by**  $\text{auto}$   
**{**  
**from**  $S\text{empty}$  **obtain**  $s$  **where**  $sS: s \in S$  **by**  $\text{auto}$   
**with**  $S$  **have**  $s \in v\text{-gs } X$  **by**  $\text{auto}$   
**then** **obtain**  $Sp$  **where**  $Sp \in X$  **and**  $sSp: s = v Sp$   
**unfolding**  $v\text{-gs}\text{-def}$  **by**  $\text{auto}$   
**hence**  $*$ :  $Sp \in U$  **using**  $\langle s \in S \rangle$  **unfolding**  $U\text{-def}$  **by**  $\text{auto}$   
**from**  $* X UX$  **have**  $le: \text{card } (v Sp) \leq l \text{ finite } (v Sp) Sp \in \mathcal{G}$   
**unfolding**  $Gl\text{-def } Gl\text{-def}$  **using**  $\text{finite}\text{-}v\mathcal{G}[of Sp]$  **by**  $\text{auto}$   
**hence**  $m: v Sp \subseteq [m]$  **by**  $(\text{intro } v\mathcal{G})$   
**have**  $Vs \subseteq v Sp$  **using**  $sS sSp$  **unfolding**  $Vs\text{-def}$  **by**  $\text{auto}$   
**with**  $\text{card}\text{-mono}[OF \langle \text{finite } (v Sp) \rangle \text{ this}] \text{ finite}\text{-subset}[OF \text{ this } \langle \text{finite } (v Sp) \rangle] le$   
 $* m$   
**have**  $\text{card } Vs \leq l U \neq \{\}$   $\text{finite } Vs Vs \subseteq [m]$  **by**  $\text{auto}$   
**}**  
**hence**  $\text{card}\text{-}Vs: \text{card } Vs \leq l$  **and**  $U\text{empty}: U \neq \{\}$   
**and**  $\text{fin}\text{-}Vs: \text{finite } Vs$  **and**  $Vsm: Vs \subseteq [m]$  **by**  $\text{auto}$

```

have vGs: v Gs  $\subseteq$  Vs unfolding Gs-def by (rule v-sameprod-subset)
have GsG: Gs  $\in$   $\mathcal{G}$  unfolding Gs-def  $\mathcal{G}$ -def
  by (intro CollectI Inter-subset sameprod-mono Vsm)
have GsGl: Gs  $\in$   $\mathcal{G}l$  unfolding  $\mathcal{G}l$ -def using GsG vGs card-Vs card-mono[OF -
vGs]
  by (simp add: fin-Vs)
hence DsDl: ?New  $\in$   $\mathcal{G}l$  using UX
  unfolding  $\mathcal{G}l$ -def  $\mathcal{G}$ -def  $\mathcal{G}l$ -def  $\mathcal{G}$ -def by auto
with X U show Y  $\subseteq$   $\mathcal{G}l$  unfolding Y by auto
from X have XD: X  $\subseteq$   $\mathcal{G}$  unfolding  $\mathcal{G}l$ -def by auto
have vplus-dsU: v-gs U = S using S(1)
  unfolding v-gs-def U-def by force
have vplus-dsXU: v-gs (X - U) = v-gs X - v-gs U
  unfolding v-gs-def U-def by auto
have card (v-gs Y) = card (v-gs (X - U  $\cup$  {?New}))
  unfolding Y by simp
also have v-gs (X - U  $\cup$  {?New}) = v-gs (X - U)  $\cup$  v-gs ({?New})
  unfolding v-gs-union ..
also have v-gs ({?New}) = {v (Gs)} unfolding v-gs-def image-comp o-def by
simp
also have card (v-gs (X - U)  $\cup$  ...)  $\leq$  card (v-gs (X - U)) + card ...
  by (rule card-Un-le)
also have ...  $\leq$  card (v-gs (X - U)) + 1 by auto
also have v-gs (X - U) = v-gs X - v-gs U by fact
also have card ... = card (v-gs X) - card (v-gs U)
  by (rule card-Diff-subset, force simp: vplus-dsU finS,
insert UX, auto simp: v-gs-def)
also have card (v-gs U) = card S unfolding vplus-dsU ..
finally show card (v-gs Y)  $\leq$  card (v-gs X) - p + 1
  using S by auto
show Y  $\neq$  {} unfolding Y using Unempty by auto
{
  fix G
  assume G  $\in$  ACC X and GPOS: G  $\in$  POS
  from this[unfolding ACC-def] POS- $\mathcal{G}$  have G: G  $\in$   $\mathcal{G}$  X  $\Vdash$  G by auto
  from this[unfolding accepts-def] obtain D :: graph where
    D: D  $\in$  X D  $\subseteq$  G by auto
  have G  $\in$  ACC Y
  proof (cases D  $\in$  Y)
    case True
      with D G show ?thesis unfolding accepts-def ACC-def by auto
    next
      case False
        with D have DU: D  $\in$  U unfolding Y by auto
        from GPOS[unfolding POS-def  $\mathcal{K}$ -def] obtain K where GK: G = (v K)  $\hat{\sim} 2$ 
card (v K) = k by auto
        from DU[unfolding U-def] have v D  $\in$  S by auto
        hence Vs  $\subseteq$  v D unfolding Vs-def by auto
        also have ...  $\subseteq$  v G

```

```

    by (intro v-mono D)
  also have ... = v K unfolding GK
    by (rule v-sameprod, unfold GK, insert k2, auto)
  finally have Gs  $\subseteq$  G unfolding Gs-def GK
    by (intro sameprod-mono)
  with D DU have D  $\in$  ?U ?New  $\subseteq$  G by (auto)
  hence Y  $\vdash$  G unfolding accepts-def Y by auto
  thus ?thesis using G by auto
qed
}
thus POS  $\cap$  ACC X  $\subseteq$  ACC Y by auto

from ex-bij-betw-nat-finite[OF finS, unfolded ‹card S = p›]
obtain Si where Si: bij-betw Si {0 ..< p} S by auto
define G where G = ( $\lambda$  i. SOME Gb. Gb  $\in$  X  $\wedge$  v Gb = Si i)
{
  fix i
  assume i < p
  with Si have SiS: Si i  $\in$  S unfolding bij-betw-def by auto
  with S have Si i  $\in$  v-gs X by auto
  hence  $\exists$  G. G  $\in$  X  $\wedge$  v G = Si i
    unfolding v-gs-def by auto
  from someI-ex[OF this]
  have (G i)  $\in$  X  $\wedge$  v (G i) = Si i
    unfolding G-def by blast
  hence G i  $\in$  X v (G i) = Si i
    G i  $\in$  U v (G i)  $\in$  S using SiS unfolding U-def
    by auto
} note G = this
have SvG: S = v ‹ G ‹ {0 ..< p} unfolding Si[unfolded bij-betw-def,
  THEN conjunct2, symmetric] image-comp o-def using G(2) by auto
have injG: inj-on G {0 ..< p}
proof (standard, goal-cases)
  case (1 i j)
  hence Si i = Si j using G[of i] G[of j] by simp
  with 1(1,2) Si show i = j
    by (metis Si bij-betw-iff-bijections)
qed
define r where r = card U
have rq: r  $\geq$  p unfolding r-def ‹card S = p›[symmetric] vplus-dsU[symmetric]
  unfolding v-gs-def
  by (rule card-image-le[OF finU])

let ?Vi =  $\lambda$  i. v (G i)
let ?Vis =  $\lambda$  i. ?Vi i - Vs
define s where s = card Vs
define si where si i = card (?Vi i) for i
define ti where ti i = card (?Vis i) for i
{

```



```

fix i
assume i: i < p
have Vs-Vi: Vs ⊆ ?Vi i using i unfolding Vs-def
  using G[OF i] unfolding SvG by auto
have finVi: finite (?Vi i)
  using G(4)[OF i] S(1) sf-precond
  by (meson finite-numbers finite-subset subset-eq)
from S(1) have G i ∈ G using G(1)[OF i] X unfolding Gl-def G-def Gl-def
by auto
hence finGi: finite (G i)
  using finite-members-G by auto
have ti: ti i = si i - s unfolding ti-def si-def s-def
  by (rule card-Diff-subset[OF fin-Vs Vs-Vi])
have size1: s ≤ si i unfolding s-def si-def
  by (intro card-mono finVi Vs-Vi)
have size2: si i ≤ l unfolding si-def using G(4)[OF i] S(1) sf-precond by
auto
note Vs-Vi finVi ti size1 size2 finGi ⟨G i ∈ G⟩
} note i-props = this
define fstt where fstt e = (SOME x. x ∈ e ∧ x ∉ Vs) for e
define sndd where sndd e = (SOME x. x ∈ e ∧ x ≠ fstt e) for e
{
  fix e :: nat set
  assume *: card e = 2 ∧ e ⊆ Vs
  from *(1) obtain x y where e: e = {x,y} x ≠ y
  by (meson card-2-iff)
  with * have ∃ x. x ∈ e ∧ x ∉ Vs by auto
  from someI-ex[OF this, folded fstt-def]
  have fst: fstt e ∈ e fstt e ∉ Vs by auto
  with * e have ∃ x. x ∈ e ∧ x ≠ fstt e
  by (metis insertCI)
  from someI-ex[OF this, folded sndd-def] have snd: sndd e ∈ e sndd e ≠ fstt e
by auto
from fst snd e have {fstt e, sndd e} = e fstt e ∉ Vs fstt e ≠ sndd e by auto
} note fstt = this
{
  fix f
  assume f ∈ ACC-cf Y - ACC-cf X
  hence fake: f ∈ ACC-cf {?New} - ACC-cf U unfolding Y ACC-cf-def
accepts-def
  Diff-iff U-def Un-iff mem-Collect-eq by blast
  hence f: f ∈ F using ACC-cf-F by auto
  hence C f ∈ NEG unfolding NEG-def by auto
  with NEG-G have Cf: C f ∈ G by auto
  from fake have f ∈ ACC-cf {?New} by auto
  from this[unfolded ACC-cf-def accepts-def] Cf
  have GsCf: Gs ⊆ C f and Cf: C f ∈ G by auto
  from fake have f ∉ ACC-cf U by auto
  from this[unfolded ACC-cf-def] Cf f have ¬ (U ⊢ C f) by auto

```

```

from this[unfolded accepts-def]
have UCf:  $D \in U \implies \neg D \subseteq C f$  for D by auto
let ?prop =  $\lambda i e. f_{stt} e \in v (G i) - V_s \wedge$ 
       $s_{ndd} e \in v (G i) \wedge e \in G i \cap ([m] \hat{\sim} 2)$ 
       $\wedge f (f_{stt} e) = f (s_{ndd} e) \wedge f (s_{ndd} e) \in [k - 1] \wedge \{f_{stt} e, s_{ndd} e\} = e$ 
define pair where pair i = (if i < p then (SOME pair. ?prop i pair) else
undefined) for i
define u where u i = f_{stt} (pair i) for i
define w where w i = s_{ndd} (pair i) for i
{
  fix i
  assume i: i < p
  from i have ?Vi i  $\in S$  unfolding SvG by auto
  hence  $V_s \subseteq ?Vi$  i unfolding Vs-def by auto
  from sameprod-mono[OF this, folded Gs-def]
  have *:  $G_s \subseteq v (G i) \hat{\sim} 2$  .
  from i have Gi:  $G i \in U$  using G[OF i] by auto
  from UCf[OF Gi] i-props[OF i] have  $\neg G i \subseteq C f$  and Gi:  $G i \in \mathcal{G}$  by auto
  then obtain edge where
    edgep: edge  $\in G i$  and edgen: edge  $\notin C f$  by auto
    from edgep Gi obtain x y where edge: edge =  $\{x, y\}$ 
    and xy:  $\{x, y\} \in [m] \hat{\sim} 2$   $\{x, y\} \subseteq [m]$  card  $\{x, y\} = 2$  unfolding G-def
binprod-def
    by force
    define a where a = f_{stt} edge
    define b where b = s_{ndd} edge
    from edgen[unfolded C-def edge] xy have id: f x = f y by simp
    from edgen GsCf edge have edgen:  $\{x, y\} \notin G_s$  by auto
    from edgen[unfolded Gs-def sameprod-altdef] xy have  $\neg \{x, y\} \subseteq V_s$  by auto
    from f_{stt}[OF  $\langle \text{card } \{x, y\} = 2 \rangle$  this, folded edge, folded a-def b-def] edge
    have a: a  $\notin V_s$  and id-ab:  $\{x, y\} = \{a, b\}$  by auto
    from id-ab id have id: f a = f b by (auto simp: doubleton-eq-iff)
    let ?pair = (a, b)
    note ab = xy[unfolded id-ab]
    from f[unfolded F-def] ab have fb: f b  $\in [k - 1]$  by auto
    note edge = edge[unfolded id-ab]
    from edgep[unfolded edge] v-mem-sub[OF  $\langle \text{card } \{a, b\} = 2 \rangle$ , of G i] id
    have ?prop i edge using edge ab a fb unfolding a-def b-def by auto
    from someI[of ?prop i, OF this] have ?prop i (pair i) using i unfolding
pair-def by auto
    from this[folded u-def w-def] edgep
    have u i  $\in v (G i) - V_s$  w i  $\in v (G i)$  pair i  $\in G i \cap [m] \hat{\sim} 2$ 
      f (u i) = f (w i) f (w i)  $\in [k - 1]$  pair i =  $\{u\ i, w\ i\}$ 
      by auto
  } note uw = this
from uw(3) have Pi: pair  $\in Pi_E \{0 ..< p\}$  G unfolding pair-def by auto
define Us where Us = u ‘  $\{0 ..< p\}$ 
define Ws where Ws =  $[m] - Us$ 
{

```

```

fix  $i$ 
assume  $i: i < p$ 
note  $uwi = uw[OF\ this]$ 
from  $uwi$  have  $ex: \exists x \in [k - 1]. f \ ' \ \{u\ i, w\ i\} = \{x\}$  by auto
  from  $uwi$  have  $*$ :  $u\ i \in [m]\ w\ i \in [m]\ \{u\ i, w\ i\} \in G\ i$  by (auto simp:
sameprod-altdef)
  have  $w\ i \notin Us$ 
  proof
    assume  $w\ i \in Us$ 
    then obtain  $j$  where  $j: j < p$  and  $wij: w\ i = u\ j$  unfolding  $Us\text{-def}$  by
auto
    with  $uwi$  have  $ij: i \neq j$  unfolding  $binprod\text{-def}$  by auto
    note  $uwj = uw[OF\ j]$ 
    from  $ij\ i\ j\ Si$ [unfolded bij-betw-def]
    have  $diff: v\ (G\ i) \neq v\ (G\ j)$  unfolding  $G(2)[OF\ i]\ G(2)[OF\ j]$  inj-on-def
by auto
    from  $uwi\ wij$  have  $uj: u\ j \in v\ (G\ i)$  by auto
    with  $\langle sunflower\ S \rangle$ [unfolded sunflower-def, rule-format]  $G(4)[OF\ i]\ G(4)[OF\ j]$ 
 $uwj(1)$  diff
    have  $u\ j \in \cap\ S$  by blast
    with  $uwj(1)$ [unfolded Vs-def] show False by simp
  qed
  with  $*$  have  $wi: w\ i \in Ws$  unfolding  $Ws\text{-def}$  by auto
  from  $uwi$  have  $wi2: w\ i \in v\ (G\ i)$  by auto
  define  $W$  where  $W = Ws \cap v\ (G\ i)$ 
  from  $G(1)[OF\ i]\ X$ [unfolded Gl-def Gl-def] i-props[OF\ i]
  have finite  $(v\ (G\ i))\ card\ (v\ (G\ i)) \leq l$  by auto
  with card-mono[OF\ this(1), of\ W] have
     $W: \text{finite}\ W\ card\ W \leq l\ W \subseteq [m] - Us$  unfolding  $W\text{-def}\ Ws\text{-def}$  by auto
  from  $wi\ wi2$  have  $wi: w\ i \in W$  unfolding  $W\text{-def}$  by auto
  from  $wi\ ex\ W\ *$  have  $\{u\ i, w\ i\} \in G\ i \wedge u\ i \in [m] \wedge w\ i \in [m] - Us \wedge f\ (u\ i) = f\ (w\ i)$  by force
  } note  $uw1 = this$ 
  have  $inj: inj\text{-on}\ u\ \{0 ..< p\}$ 
  proof -
  {
    fix  $i\ j$ 
    assume  $i: i < p$  and  $j: j < p$ 
      and  $id: u\ i = u\ j$  and  $ij: i \neq j$ 
    from  $ij\ i\ j\ Si$ [unfolded bij-betw-def]
    have  $diff: v\ (G\ i) \neq v\ (G\ j)$  unfolding  $G(2)[OF\ i]\ G(2)[OF\ j]$  inj-on-def
by auto
    from  $uw[OF\ i]$  have  $ui: u\ i \in v\ (G\ i) - Vs$  by auto
    from  $uw[OF\ j, folded\ id]$  have  $uj: u\ i \in v\ (G\ j)$  by auto
    with  $\langle sunflower\ S \rangle$ [unfolded sunflower-def, rule-format]  $G(4)[OF\ i]\ G(4)[OF\ j]$ 
 $uw[OF\ i]$  diff
    have  $u\ i \in \cap\ S$  by blast
    with  $ui$  have False unfolding  $Vs\text{-def}$  by auto
  }

```

```

thus ?thesis unfolding inj-on-def by fastforce
qed
have card: card ([m] - Us) = m - p
proof (subst card-Diff-subset)
  show finite Us unfolding Us-def by auto
  show Us  $\subseteq$  [m] unfolding Us-def using uw1 by auto
  have card Us = p unfolding Us-def using inj
    by (simp add: card-image)
  thus card [m] - card Us = m - p by simp
qed
hence ( $\forall i < p$ . pair i  $\in$  G i)  $\wedge$  inj-on u {0 ..< p}  $\wedge$  ( $\forall i < p$ . w i  $\in$  [m] - u
‘ {0 ..< p}  $\wedge$  f (u i) = f (w i))
  using inj uw1 uw unfolding Us-def by auto
  from this[unfolded u-def w-def] Pi card[unfolded Us-def u-def w-def]
  have  $\exists e \in Pi_E$  {0..<p} G. ( $\forall i < p$ . e i  $\in$  G i)  $\wedge$ 
    card ([m] - ( $\lambda i$ . fstt (e i)) ‘ {0..<p}) = m - p  $\wedge$ 
    ( $\forall i < p$ . sndd (e i)  $\in$  [m] - ( $\lambda i$ . fstt (e i)) ‘ {0..<p})  $\wedge$  f (fstt (e i)) = f (sndd
(e i))
    by blast
  } note fMem = this
define Pi2 where Pi2 W = Pi_E ([m] - W) ( $\lambda \_$ . [k - 1]) for W
define merge where merge =
  ( $\lambda e$  (g :: nat  $\Rightarrow$  nat) v. if v  $\in$  ( $\lambda i$ . fstt (e i)) ‘ {0 ..< p} then g (sndd (e
(SOME i. i < p  $\wedge$  v = fstt (e i)))) else g v)
  let ?W =  $\lambda e$ . ( $\lambda i$ . fstt (e i)) ‘ {0..<p}
  have ACC-cf Y - ACC-cf X  $\subseteq$  { merge e g | e g. e  $\in$  Pi_E {0..<p} G  $\wedge$  card
([m] - ?W e) = m - p  $\wedge$  g  $\in$  Pi2 (?W e)}
  (is -  $\subseteq$  ?R)
proof
  fix f
  assume mem: f  $\in$  ACC-cf Y - ACC-cf X
  with ACC-cf-F have f  $\in$  F by auto
  hence f: f  $\in$  [m]  $\rightarrow_E$  [k - 1] unfolding F-def .
  from fMem[OF mem] obtain e where e: e  $\in$  Pi_E {0..<p} G
     $\wedge$  i. i < p  $\implies$  e i  $\in$  G i
    card ([m] - ?W e) = m - p
     $\wedge$  i. i < p  $\implies$  sndd (e i)  $\in$  [m] - ?W e  $\wedge$  f (fstt (e i)) = f (sndd (e i)) by auto
  define W where W = ?W e
  note e = e[folded W-def]
  let ?g = restrict f ([m] - W)
  let ?h = merge e ?g
  have f  $\in$  ?R
  proof (intro CollectI exI[of - e] exI[of - ?g], unfold W-def[symmetric], intro
conjI e)
    show ?g  $\in$  Pi2 W unfolding Pi2-def using f by auto
    {
      fix v :: nat
      have ?h v = f v
      proof (cases v  $\in$  W)

```

**case** *False*  
**thus** *?thesis* **using** *f unfolding merge-def unfolding W-def[symmetric]*  
**by** *auto*  
**next**  
**case** *True*  
**from** *this[unfolded W-def]* **obtain** *i* **where** *i: i < p and v: v = fstt (e i)*  
**by** *auto*  
**define** *j* **where** *j = (SOME j. j < p ∧ v = fstt (e j))*  
**from** *i v* **have**  $\exists j. j < p \wedge v = \text{fstt } (e j)$  **by** *auto*  
**from** *someI-ex[OF this, folded j-def]* **have** *j: j < p and v: v = fstt (e j)*  
**by** *auto*  
**have** *?h v = restrict f ([m] - W) (sndd (e j))*  
**unfolding** *merge-def unfolding W-def[symmetric] j-def* **using** *True* **by**  
*auto*  
**also** **have**  $\dots = f (\text{sndd } (e j))$  **using** *e(4)[OF j]* **by** *auto*  
**also** **have**  $\dots = f (\text{fstt } (e j))$  **using** *e(4)[OF j]* **by** *auto*  
**also** **have**  $\dots = f v$  **using** *v* **by** *simp*  
**finally** **show** *?thesis* .  
**qed**  
**}**  
**thus** *f = ?h* **by** *auto*  
**qed**  
**thus** *f ∈ ?R* **by** *auto*  
**qed**  
**also** **have**  $\dots \subseteq (\lambda (e,g). (\text{merge } e g)) \text{ ' } (\text{Sigma } (Pi_E \{0..<p\} G \cap \{e. \text{card } ([m] - ?W e) = m - p\}) (\lambda e. Pi2 (?W e)))$   
*(is - ⊆ ?f ' ?R)*  
**by** *auto*  
**finally** **have** *sub: ACC-cf Y - ACC-cf X ⊆ ?f ' ?R* .  
**have** *fin[simp,intro]: finite [m] finite [k - Suc 0] unfolding numbers-def* **by** *auto*  
**have** *finPie[simp, intro]: finite (Pi\_E {0..<p} G)*  
**by** *(intro finite-PiE, auto intro: i-props)*  
**have** *finR: finite ?R unfolding Pi2-def*  
**by** *(intro finite-SigmaI finite-Int allI finite-PiE i-props, auto)*  
**have** *card (ACC-cf Y - ACC-cf X) ≤ card (?f ' ?R)*  
**by** *(rule card-mono[OF finite-imageI[OF finR] sub])*  
**also** **have**  $\dots \leq \text{card } ?R$   
**by** *(rule card-image-le[OF finR])*  
**also** **have**  $\dots = (\sum e \in (Pi_E \{0..<p\} G \cap \{e. \text{card } ([m] - ?W e) = m - p\}). \text{card } (Pi2 (?W e)))$   
**by** *(rule card-SigmaI, unfold Pi2-def,*  
*(intro finite-SigmaI allI finite-Int finite-PiE i-props, auto)+)*  
**also** **have**  $\dots = (\sum e \in Pi_E \{0..<p\} G \cap \{e. \text{card } ([m] - ?W e) = m - p\}. (k - 1) \wedge (\text{card } ([m] - ?W e)))$   
**by** *(rule sum.cong[OF refl], unfold Pi2-def, subst card-PiE, auto)*  
**also** **have**  $\dots = (\sum e \in Pi_E \{0..<p\} G \cap \{e. \text{card } ([m] - ?W e) = m - p\}. (k - 1) \wedge (m - p))$   
**by** *(rule sum.cong[OF refl], rule arg-cong[of - - λ n. (k - 1) ^ n], auto)*  
**also** **have**  $\dots \leq (\sum e \in Pi_E \{0..<p\} G. (k - 1) \wedge (m - p))$

by (rule sum-mono2, auto)  
 also have ... = card (Pi<sub>E</sub> {0..<sub>p</sub>} G) \* (k - 1) ^ (m - p) by simp  
 also have ... = (∏ i = 0..<sub>p</sub>. card (G i)) \* (k - 1) ^ (m - p)  
 by (subst card-PiE, auto)  
 also have ... ≤ (∏ i = 0..<sub>p</sub>. (k - 1) div 2) \* (k - 1) ^ (m - p)  
 proof -  
 {  
 fix i  
 assume i: i < p  
 from G[OF i] X  
 have GiG: G i ∈ G  
 unfolding G1-def G-def G-def sameprod-altdef by force  
 from i-props[OF i] have finGi: finite (G i) by auto  
 have finvGi: finite (v (G i)) by (rule finite-vG, insert i-props[OF i], auto)  
 have card (G i) ≤ card ((v (G i))<sup>2</sup>)  
 by (intro card-mono[OF sameprod-finite], rule finvGi, rule v-G-2[OF GiG])  
 also have ... ≤ l choose 2  
 proof (subst card-sameprod[OF finvGi], rule choose-mono)  
 show card (v (G i)) ≤ l using i-props[OF i] unfolding ti-def si-def by  
 simp  
 qed  
 also have l choose 2 = l \* (l - 1) div 2 unfolding choose-two by simp  
 also have l \* (l - 1) = k - l unfolding kl2 power2-eq-square by (simp add:  
 algebra-simps)  
 also have ... div 2 ≤ (k - 1) div 2  
 by (rule div-le-mono, insert l2, auto)  
 finally have card (G i) ≤ (k - 1) div 2 .  
 }  
 thus ?thesis by (intro mult-right-mono prod-mono, auto)  
 qed  
 also have ... = ((k - 1) div 2) ^ p \* (k - 1) ^ (m - p)  
 by simp  
 also have ... ≤ ((k - 1) ^ p div (2<sup>p</sup>)) \* (k - 1) ^ (m - p)  
 by (rule mult-right-mono; auto simp: div-mult-pow-le)  
 also have ... ≤ ((k - 1) ^ p \* (k - 1) ^ (m - p)) div 2<sup>p</sup>  
 by (rule div-mult-le)  
 also have ... = (k - 1) ^ m div 2<sup>p</sup>  
 proof -  
 have p + (m - p) = m using mp by simp  
 thus ?thesis by (subst power-add[symmetric], simp)  
 qed  
 finally have card (ACC-cf Y - ACC-cf X) ≤ (k - 1) ^ m div 2<sup>p</sup> .  
 hence 2<sup>p</sup> \* card (ACC-cf Y - ACC-cf X) ≤ 2<sup>p</sup> \* ((k - 1) ^ m div 2<sup>p</sup>)  
 by simp  
 also have ... ≤ (k - 1) ^ m by simp  
 finally show 2<sup>p</sup> \* card (ACC-cf Y - ACC-cf X) ≤ (k - 1) ^ m .  
 qed

Definition 6

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

**termination**

**proof** (*relation measure* ( $\lambda X. \text{card} (v\text{-gs } X)$ ), *force*, *goal-cases*)  
**case** (1 *X*)  
**hence**  $X \subseteq \mathcal{G}l$  **and** *LL*:  $L < \text{card} (v\text{-gs } X)$  **by** *auto*  
**from** *plucking-step*(1)[*OF this refl*]  
**have**  $\text{card} (v\text{-gs } (\text{plucking-step } X)) \leq \text{card} (v\text{-gs } X) - p + 1$  .  
**also have**  $\dots < \text{card} (v\text{-gs } X)$  **using** *p L3 LL*  
**by** *auto*  
**finally show** ?*case* **by** *simp*  
**qed**

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

**definition** *PLU* :: *graph set*  $\Rightarrow$  *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**  $n * (p - 1) \leq \text{card} (v\text{-gs } X)$   
**using** *assms*  
**proof** (*induct* *X* *arbitrary*: *Z* *n* *rule*: *PLU-main.induct*)  
**case** (1 *X* *Z* *n*)  
**note** [*simp*] = *PLU-main.simps*[*of* *X*]  
**show** ?*case*  
**proof** (*cases*  $\text{card} (v\text{-gs } X) \leq L$ )  
**case** *True*  
**thus** ?*thesis* **using** 1 **by** *auto*  
**next**  
**case** *False*  
**define** *Y* **where** *Y* = *plucking-step* *X*  
**obtain** *q* **where** *PLU*: *PLU-main* *Y* = (*Z*, *q*) **and** *n*:  $n = \text{Suc } q$   
**using**  $\langle \text{PLU-main } X = (Z, n) \rangle$ [*unfolded PLU-main.simps*[*of* *X*], *folded Y-def*]  
**using** *False* 1(2) **by** (*cases* *PLU-main* *Y*, *auto*)  
**from** *False* **have** *L*:  $\text{card} (v\text{-gs } X) > L$  **by** *auto*  
**note** *step* = *plucking-step*[*OF* 1(2) *this Y-def*]  
**from** *False* 1 **have**  $X \subseteq \mathcal{G}l \wedge L < \text{card} (v\text{-gs } X)$  **by** *auto*  
**note** *IH* = 1(1)[*folded Y-def*, *OF this step*(2) *PLU*]  
**have**  $n * (p - 1) = (p - 1) + q * (p - 1)$  **unfolding** *n* **by** *simp*  
**also have**  $\dots \leq (p - 1) + \text{card} (v\text{-gs } Y)$  **using** *IH* **by** *simp*  
**also have**  $\dots \leq p - 1 + (\text{card} (v\text{-gs } X) - p + 1)$  **using** *step*(1) **by** *simp*  
**also have**  $\dots = \text{card} (v\text{-gs } X)$  **using** *L Lp p* **by** *simp*  
**finally show** ?*thesis* .  
**qed**

qed

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**  $X \subseteq \mathcal{G}l$

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

**shows**  $Z \in \mathcal{P}LGl$

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

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

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

**using** *assms*

**proof** (*induct*  $X$  *arbitrary*:  $Z n$  *rule*: *PLU-main.induct*)

**case** ( $1 X Z n$ )

**note** [*simp*] = *PLU-main.simps*[of  $X$ ]

**show** ?*case*

**proof** (*cases*  $\text{card} (v\text{-}gs X) \leq L$ )

**case** *True*

**from** *True* **show** ?*thesis* **using** 1 **by** (*auto simp: id PLGgl-def*)

**next**

**case** *False*

**define**  $Y$  **where**  $Y = \text{plucking-step } X$

**obtain**  $q$  **where** *PLU*: *PLU-main*  $Y = (Z, q)$  **and**  $n: n = \text{Suc } q$

**using**  $\langle \text{PLU-main } X = (Z, n) \rangle$  [*unfolded PLU-main.simps*[of  $X$ ], *folded Y-def*]

**using** *False* 1(2) **by** (*cases PLU-main Y, auto*)

**from** *False* **have**  $\text{card} (v\text{-}gs X) > L$  **by** *auto*

**note**  $\text{step} = \text{plucking-step}$  [*OF* 1(2) *this Y-def*]

**from** *False* 1 **have**  $X \subseteq \mathcal{G}l \wedge L < \text{card} (v\text{-}gs X)$  **by** *auto*

**note**  $IH = 1(1)$  [*folded Y-def, OF this step(2) PLU*]  $\langle Y \neq \{\} \rangle$

**let** ?*Diff* =  $\lambda X Y. ACC\text{-}cf X - ACC\text{-}cf Y$

**have** *finNEG*: *finite NEG*

**using** *NEG-G infinite-super* **by** *blast*



**have**  $?Diff\ Z\ X \subseteq ?Diff\ Z\ Y \cup ?Diff\ Y\ X$  **by** *auto*  
**from** *card-mono[OF finite-subset[OF - finite- $\mathcal{F}$ ] this] ACC-cf- $\mathcal{F}$*   
**have**  $2^{\wedge p} * card\ (?Diff\ Z\ X) \leq 2^{\wedge p} * card\ (?Diff\ Z\ Y \cup ?Diff\ Y\ X)$  **by** *auto*  
**also have**  $\dots \leq 2^{\wedge p} * (card\ (?Diff\ Z\ Y) + card\ (?Diff\ Y\ X))$   
**by** (*rule mult-left-mono, rule card-Un-le, simp*)  
**also have**  $\dots = 2^{\wedge p} * card\ (?Diff\ Z\ Y) + 2^{\wedge p} * card\ (?Diff\ Y\ X)$   
**by** (*simp add: algebra-simps*)  
**also have**  $\dots \leq ((k - 1)^{\wedge m}) * q + (k - 1)^{\wedge m}$  **using** *IH step* **by** *auto*  
**also have**  $\dots = ((k - 1)^{\wedge m}) * Suc\ q$  **by** (*simp add: ac-simps*)  
**finally have**  $c: 2^{\wedge p} * card\ (ACC-cf\ Z - ACC-cf\ X) \leq ((k - 1)^{\wedge m}) * Suc\ q$  **by** *simp*  
**from** *False* **have**  $X \neq \{\}$  **by** *auto*  
**thus** *?thesis* **unfolding**  $n$  **using** *IH step c* **by** *auto*  
**qed**  
**qed**

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

**proof** –

**obtain**  $Z\ n$  **where** *res*:  $PLU-main\ (X \cup Y) = (Z, n)$  **by** *force*

**hence** *PLU*:  $PLU\ (X \cup Y) = Z$  **unfolding** *PLU-def* **by** *simp*

**from**  $X\ Y$  **have** *XY*:  $X \cup Y \subseteq Gl$  **unfolding** *PLGl-def* **by** *auto*

**note** *main* =  $PLU-main$ [*OF this(1) res*]

**from** *main* **show**  $PLU\ (X \cup Y) \in \mathcal{PLGl}$  **unfolding** *PLU* **by** *simp*

**thus**  $X \sqcup Y \in \mathcal{PLGl}$  **unfolding** *sqcup-def* .

**from** *main* **show**  $POS \cap ACC\ (X \cup Y) \subseteq ACC\ (X \sqcup Y)$

**unfolding** *sqcup-def PLU* **by** *simp*

**thus**  $\partial \sqcup Pos\ X\ Y = \{\}$  **unfolding** *deviate-pos-cup-def PLU sqcup-def* **by** *auto*

**have**  $card\ (v-gs\ (X \cup Y)) \leq card\ (v-gs\ X) + card\ (v-gs\ Y)$

**unfolding** *v-gs-union* **by** (*rule card-Un-le*)

**also have**  $\dots \leq L + L$  **using**  $X\ Y$  **unfolding** *PLGl-def* **by** *simp*

**finally have**  $card\ (v-gs\ (X \cup Y)) \leq 2 * L$  **by** *simp*

**with** *PLU-main-n*[*OF XY(1) res*] **have**  $n * (p - 1) \leq 2 * L$  **by** *simp*

**with**  $p\ Lm\ m2$  **have**  $n: n < 2 * L$  **by** (*cases n, auto, cases p - 1, auto*)

**let**  $?r = real$

**have**  $*$ :  $(k - 1)^{\wedge m} > 0$  **using**  $k\ l2$  **by** *simp*

**have**  $2^{\wedge p} * card\ (\partial \sqcup Neg\ X\ Y) \leq 2^{\wedge p} * card\ (ACC-cf\ Z - ACC-cf\ (X \cup Y))$

**unfolding** *deviate-neg-cup-def PLU sqcup-def*

**by** (*rule mult-left-mono, rule card-mono[OF finite-subset[OF - finite- $\mathcal{F}$ ]], insert ACC-cf- $\mathcal{F}$ , force, auto*)

**also have**  $\dots \leq (k - 1)^{\wedge m} * n$  **using** *main* **by** *simp*

**also have**  $\dots < (k - 1)^{\wedge m} * (2 * L)$  **unfolding** *mult-less-cancel1* **using**  $n *$

**by** *simp*

**also have**  $\dots = 2 * ((k - 1)^{\wedge m} * L)$  **by** *simp*

**finally have**  $2 * (2^{p-1} * \text{card}(\partial \sqcup \text{Neg } X Y)) < 2 * ((k-1)^m * L)$   
**using**  $p$  **by** (*cases p, auto*)  
**hence**  $2^{p-1} * \text{card}(\partial \sqcup \text{Neg } X Y) < (k-1)^m * L$  **by** *simp*  
**hence**  $?r (2^{p-1} * \text{card}(\partial \sqcup \text{Neg } X Y)) < ?r ((k-1)^m * L)$  **by** *linarith*  
**thus**  $\text{card}(\partial \sqcup \text{Neg } X Y) < (k-1)^m * L / 2^{p-1}$  **by** (*simp add: field-simps*)  
**qed**

Lemma 10

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

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

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

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

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

**proof** –

**obtain**  $Z n$  **where** *res*:  $PLU\text{-main}(X \odot Y) = (Z, n)$  **by** *force*

**hence** *PLU*:  $PLU(X \odot Y) = Z$  **unfolding** *PLU-def* **by** *simp*

**from**  $X Y$  **have**  $XY: X \subseteq \mathcal{Gl} Y \subseteq \mathcal{Gl} X \subseteq \mathcal{G} Y \subseteq \mathcal{G}$  **unfolding** *PLGl-def Gl-def*

**by** *auto*

**have** *sub*:  $X \odot Y \subseteq \mathcal{Gl}$  **unfolding** *odotl-def* **using**  $XY$

**by** (*auto split: option.splits*)

**note** *main* =  $PLU\text{-main}[OF \text{ sub } res]$

**note** *finv* =  $\text{finite-v-gs-Gl}[OF XY(1)] \text{ finite-v-gs-Gl}[OF XY(2)]$

**have**  $X \odot Y \subseteq \mathcal{G}$  **by** (*rule odot-G, insert XY, auto simp: Gl-def*)

**hence**  $XYD: X \odot Y \subseteq \mathcal{G}$  **by** *auto*

**have** *finvXY*:  $\text{finite}(v\text{-gs}(X \odot Y))$  **by** (*rule finite-v-gs[OF XYD]*)

**have**  $\text{card}(v\text{-gs}(X \odot Y)) \leq \text{card}(v\text{-gs } X) * \text{card}(v\text{-gs } Y)$

**using**  $XY(1-2)$  **by** (*intro card-v-gs-join, auto simp: Gl-def*)

**also have**  $\dots \leq L * L$  **using**  $XY$  **unfolding** *PLGl-def*

**by** (*intro mult-mono, auto*)

**also have**  $\dots = L^2$  **by** *algebra*

**finally have** *card-join*:  $\text{card}(v\text{-gs}(X \odot Y)) \leq L^2$  .

**with** *card-mono*[ $OF \text{ finvXY } v\text{-gs-mono}[OF \text{ joinl-join}]$ ]

**have** *card*:  $\text{card}(v\text{-gs}(X \odot Y)) \leq L^2$  **by** *simp*

**with** *PLU-main-n*[ $OF \text{ sub } res$ ] **have**  $n * (p-1) \leq L^2$  **by** *simp*

**with**  $p \text{ Lm } m2$  **have**  $n: n < 2 * L^2$  **by** (*cases n, auto, cases p-1, auto*)

**have**  $*(k-1)^m > 0$  **using**  $k \text{ l2}$  **by** *simp*

**show**  $PLU(X \odot Y) \in \mathcal{PLGl}$  **unfolding** *PLU* **using** *main* **by** *auto*

**thus**  $X \sqcap Y \in \mathcal{PLGl}$  **unfolding** *sqcap-def* .

**let**  $?r = \text{real}$

**have**  $2^p * \text{card}(\partial \sqcap \text{Neg } X Y) \leq 2^p * \text{card}(ACC\text{-cf } Z - ACC\text{-cf}(X \odot Y))$

**unfolding** *deviate-neg-cap-def PLU sqcap-def*

**by** (*rule mult-left-mono, rule card-mono[OF finite-subset[OF - finite-F]], insert ACC-cf-F, force,*

*insert ACC-cf-mono[OF joinl-join, of X Y], auto*)

**also have**  $\dots \leq (k-1)^m * n$  **using** *main* **by** *simp*

**also have**  $\dots < (k-1)^m * (2 * L^2)$  **unfolding** *mult-less-cancel1* **using**  $n$  **by** *simp*

**finally have**  $2 * (2^{p-1} * \text{card}(\partial \sqcap \text{Neg } X Y)) < 2 * ((k-1)^m * L^2)$

**using**  $p$  **by** (*cases p, auto*)

**hence**  $2 \wedge (p - 1) * \text{card} (\partial \sqcap \text{Neg } X \ Y) < (k - 1) \wedge m * L \wedge 2$  **by** *simp*  
**hence**  $?r (2 \wedge (p - 1) * \text{card} (\partial \sqcap \text{Neg } X \ Y)) < (k - 1) \wedge m * L \wedge 2$  **by** *linarith*  
**thus**  $\text{card} (\partial \sqcap \text{Neg } X \ Y) < (k - 1) \wedge m * L \wedge 2 / 2 \wedge (p - 1)$  **by** (*simp add: field-simps*)

**define** *Vs* **where**  $Vs = v\text{-gs } (X \odot Y) \cap \{V . V \subseteq [m] \wedge \text{card } V \geq \text{Suc } l\}$   
**define** *C* **where**  $C (V :: \text{nat set}) = (\text{SOME } C . C \subseteq V \wedge \text{card } C = \text{Suc } l)$  **for** *V*  
**define** *K* **where**  $K \ C = \{W . W \subseteq [m] - C \wedge \text{card } W = k - \text{Suc } l\}$  **for** *C*  
**define** *merge* **where**  $\text{merge } C \ V = (C \cup V) \wedge 2$  **for**  $C \ V :: \text{nat set}$   
**define** *GS* **where**  $GS = \{\text{merge } (C \ V) \ W \mid V \ W . V \in Vs \wedge W \in K (C \ V)\}$   
{  
  **fix** *V*  
  **assume**  $V : V \in Vs$   
  **hence**  $\text{card } V \geq \text{Suc } l$  **and**  $Vm : V \subseteq [m]$  **unfolding** *Vs-def* **by** *auto*  
  **from** *card* **obtain** *D* **where**  $C : D \subseteq V$  **and**  $\text{card } D = \text{Suc } l$   
  **by** (*rule obtain-subset-with-card-n*)  
  **hence**  $\exists C . C \subseteq V \wedge \text{card } C = \text{Suc } l$  **by** *blast*  
  **from** *someI-ex*[*OF this, folded C-def*] **have**  $*$ :  $C \ V \subseteq V$   $\text{card } (C \ V) = \text{Suc } l$   
  **by** *blast+*  
  **with** *Vm* **have**  $\text{sub} : C \ V \subseteq [m]$  **by** *auto*  
  **from** *finite-subset*[*OF this*] **have** *finCV*:  $\text{finite } (C \ V)$  **unfolding** *numbers-def*  
**by** *simp*  
  **have**  $\text{card } (K (C \ V)) = (m - \text{Suc } l)$  *choose*  $(k - \text{Suc } l)$  **unfolding** *K-def*  
  **proof** (*subst n-subsets, (rule finite-subset[of - [m]], auto)[1], rule arg-cong[of -*  
-  $\lambda x . x$  *choose -]*)  
  **show**  $\text{card } ([m] - C \ V) = m - \text{Suc } l$   
  **by** (*subst card-Diff-subset, insert sub \* finCV, auto*)  
  **qed**  
  **note**  $*$  *finCV sub this*  
} **note** *Vs-C = this*  
**have** *finK*:  $\text{finite } (K \ V)$  **for** *V* **unfolding** *K-def* **by** *auto*  
{  
  **fix** *G*  
  **assume**  $G : G \in \text{POS} \cap \text{ACC } (X \odot Y)$   
  **have**  $G \in \text{ACC } (X \odot l \ Y) \cup GS$   
  **proof** (*rule ccontr*)  
  **assume**  $\neg ?thesis$   
  **with** *G* **have**  $G : G \in \text{POS} \ G \in \text{ACC } (X \odot Y) \ G \notin \text{ACC } (X \odot l \ Y)$   
  **and** *contra*:  $G \notin GS$  **by** *auto*  
  **from**  $G(1)$ [*unfolded K-def*] **have**  $\text{card } (v \ G) = k \wedge (v \ G) \wedge 2 = G$  **and**  $G0 : G$   
 $\in \mathcal{G}$   
  **by** *auto*  
  **hence**  $vGk : \text{card } (v \ G) = k \ (v \ G) \wedge 2 = G$  **by** *auto*  
  **from** *G0* **have**  $vm : v \ G \subseteq [m]$  **by** (*rule v-G*)  
  **from**  $G(2-3)$ [*unfolded ACC-def accepts-def*] **obtain** *H*  
  **where**  $H : H \in X \odot Y \ H \notin X \odot l \ Y$   
  **and**  $HG : H \subseteq G$  **by** *auto*  
  **from** *v-mono*[*OF HG*] **have**  $vHG : v \ H \subseteq v \ G$  **by** *auto*  
  {  
  **from**  $H(1)$ [*unfolded odot-def*] **obtain** *D E* **where**  $D : D \in X$  **and**  $E : E \in$

**Y and HDE:  $H = D \cup E$**   
 by force  
 from  $D E X Y$  have  $Dl: D \in \mathcal{G}l E \in \mathcal{G}l$  unfolding  $PL\mathcal{G}l\text{-def}$  by auto  
 have  $Dp: D \in \mathcal{G}$  using  $Dl$  by (auto simp:  $\mathcal{G}l\text{-def}$ )  
 have  $Ep: E \in \mathcal{G}$  using  $Dl$  by (auto simp:  $\mathcal{G}l\text{-def}$ )  
 from  $Dl HDE$  have  $HD: H \in \mathcal{G}$  unfolding  $\mathcal{G}l\text{-def}$  by auto  
 have  $HG0: H \in \mathcal{G}$  using  $Dp Ep$  unfolding  $HDE$  by auto  
 have  $HDL: H \notin \mathcal{G}l$   
 proof  
 assume  $H \in \mathcal{G}l$   
 hence  $H \in X \odot l Y$   
 unfolding  $odotl\text{-def HDE odot-def}$  using  $D E$  by blast  
 thus False using  $H$  by auto  
 qed  
 from  $HDL HD$  have  $HGl: H \notin \mathcal{G}l$  unfolding  $\mathcal{G}l\text{-def}$  by auto  
 have  $vm: v H \subseteq [m]$  using  $HG0$  by (rule  $v\text{-}\mathcal{G}$ )  
 have lower:  $l < \text{card } (v H)$  using  $HGl HG0$  unfolding  $\mathcal{G}l\text{-def}$  by auto  
 have  $v H \in Vs$  unfolding  $Vs\text{-def}$  using lower  $vm H$  unfolding  $v\text{-}gs\text{-def}$   
 by auto  
 } note  $in\text{-}Vs = \text{this}$   
 note  $C = Vs\text{-}C[OF \text{this}]$   
 let  $?C = C (v H)$   
 from  $C vHG$  have  $CG: ?C \subseteq v G$  by auto  
 hence  $id: v G = ?C \cup (v G - ?C)$  by auto  
 from  $arg\text{-}cong[OF \text{this}, of \text{card}] vGk(1) C$   
 have  $\text{card } (v G - ?C) = k - \text{Suc } l$   
 by (metis  $CG \text{card-Diff-subset}$ )  
 hence  $v G - ?C \in K ?C$  unfolding  $K\text{-def}$  using  $vm$  by auto  
 hence  $\text{merge } ?C (v G - ?C) \in GS$  unfolding  $GS\text{-def}$  using  $in\text{-}Vs$  by auto  
 also have  $\text{merge } ?C (v G - ?C) = v G \text{ } \mathbf{2}$  unfolding  $\text{merge-def}$   
 by (rule  $arg\text{-}cong[of - - \text{sameprod}], \text{insert id}, \text{auto}$ )  
 also have  $\dots = G$  by fact  
 finally have  $G \in GS$  .  
 with  $\text{contra show False ..}$   
 qed  
 }  
 hence  $\partial \sqcap Pos X Y \subseteq (POS \cap ACC (X \odot l Y) - ACC (X \sqcap Y)) \cup GS$   
 unfolding  $\text{deviate-pos-cap-def}$  by auto  
 also have  $POS \cap ACC (X \odot l Y) - ACC (X \sqcap Y) = \{\}$   
 proof -  
 have  $POS - ACC (X \sqcap Y) \subseteq UNIV - ACC (X \odot l Y)$   
 unfolding  $\text{sqcap-def}$  using  $PLU \text{main}$  by auto  
 thus  $?thesis$  by auto  
 qed  
 finally have  $\text{sub: } \partial \sqcap Pos X Y \subseteq GS$  by auto  
 have  $finVs: \text{finite } Vs$  unfolding  $Vs\text{-def numbers-def}$  by simp  
 let  $?Sig = \text{Sigma } Vs (\lambda V. K (C V))$   
 have  $GS\text{-def: } GS = (\lambda (V, W). \text{merge } (C V) W) \text{ } \text{' } ?Sig$  unfolding  $GS\text{-def}$   
 by auto

```

have finSig: finite ?Sig using finVs finK by simp
have finGS: finite GS unfolding GS-def
  by (rule finite-imageI[OF finSig])
have card ( $\partial \Pi Pos X Y$ )  $\leq$  card GS by (rule card-mono[OF finGS sub])
also have  $\dots \leq$  card ?Sig unfolding GS-def
  by (rule card-image-le[OF finSig])
also have  $\dots =$  ( $\sum a \in Vs. \text{card } (K (C a))$ )
  by (rule card-SigmaI[OF finVs], auto simp: finK)
also have  $\dots =$  ( $\sum a \in Vs. (m - Suc l) \text{ choose } (k - Suc l)$ ) using Vs-C
  by (intro sum.cong, auto)
also have  $\dots =$  ( $(m - Suc l) \text{ choose } (k - Suc l) * \text{card } Vs$ )
  by simp
also have  $\dots \leq$  ( $(m - Suc l) \text{ choose } (k - Suc l) * L^2$ )
proof (rule mult-left-mono)
  have card Vs  $\leq$  card (v-gs ( $X \odot Y$ ))
    by (rule card-mono[OF finvXY], auto simp: Vs-def)
  also have  $\dots \leq L^2$  by fact
  finally show card Vs  $\leq L^2$  .
qed simp
finally show card ( $\partial \Pi Pos X Y$ )  $\leq$  ( $(m - l - 1) \text{ choose } (k - l - 1) * L^2$ )
  by simp
qed
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: card  $\mathcal{V} = (m \text{ choose } 2)$ 
  and bij-betw- $\pi$ : bij-betw  $\pi \mathcal{V} ([m]^2)$ 
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 A-simps[simp]:

```

```

  FALSE  $\in \mathcal{A}$ 
  (Var  $x \in \mathcal{A}$ ) = ( $x \in \mathcal{V}$ )
  (Conj  $\varphi \psi \in \mathcal{A}$ ) = ( $\varphi \in \mathcal{A} \wedge \psi \in \mathcal{A}$ )
  (Disj  $\varphi \psi \in \mathcal{A}$ ) = ( $\varphi \in \mathcal{A} \vee \psi \in \mathcal{A}$ )
  by (auto simp: A-def)

```

```

lemma inj-on- $\pi$ : inj-on  $\pi \mathcal{V}$ 
  using bij-betw- $\pi$  by (metis bij-betw-imp-inj-on)

```

**lemma**  $\pi m2[simp,intro]: x \in \mathcal{V} \implies \pi x \in [m] \hat{=} 2$   
**using** *bij-betw- $\pi$*  **by** (*rule bij-betw-apply*)

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

**proof** –

**from**  $\pi m2[OF \text{assms}]$  **have** *mem*:  $\pi x \in [m] \hat{=} 2$  **by** *auto*

**from** *this*[*unfolded binprod-def*] **obtain**  $a b$  **where**  $\pi: \pi x = \{a,b\}$  **and** *diff*:  $a \neq b$

**by** *auto*

**hence**  $v \{\pi x\} = \{a,b\}$  **unfolding** *v-def* **by** *auto*

**thus** *?thesis* **using** *diff* **by** *simp*

**qed**

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

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

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

**using** *assms L3 l2*

**by** (*auto simp:  $\mathcal{G}$ -def  $\mathcal{PLGl}$ -def v-gs-def  $\mathcal{Gl}$ -def*)

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

**by** (*auto simp:  $\mathcal{G}$ -def  $\mathcal{PLGl}$ -def v-gs-def  $\mathcal{Gl}$ -def*)

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

*SET* *FALSE* =  $\{\}$

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

| *SET* (*Disj*  $\varphi \psi$ ) = *SET*  $\varphi \cup$  *SET*  $\psi$

| *SET* (*Conj*  $\varphi \psi$ ) = *SET*  $\varphi \odot$  *SET*  $\psi$

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

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

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

*ACC-cf* (*SET* (*Disj*  $\varphi \psi$ )) = *ACC-cf* (*SET*  $\varphi$ )  $\cup$  *ACC-cf* (*SET*  $\psi$ )

*ACC-cf* (*SET* (*Conj*  $\varphi \psi$ )) = *ACC-cf* (*SET*  $\varphi$ )  $\cap$  *ACC-cf* (*SET*  $\psi$ )

**using** *ACC-cf-odot*

**by** (*auto simp: ACC-cf-union ACC-cf-empty, auto simp: ACC-cf-def accepts-def*)

**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*  $\psi$ )

**by** (*auto simp: ACC-union ACC-odot, auto simp: ACC-def accepts-def*)

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

**proof** (*induct  $\varphi$  rule: tf-mformula.induct*)

**case** (*tf-Conj*  $\varphi \psi$ )

**hence** *SET*  $\varphi \subseteq \mathcal{G}$  *SET*  $\psi \subseteq \mathcal{G}$  **by** *auto*

**from** *odot- $\mathcal{G}$* [*OF this*] **show** *?case* **by** *simp*

**qed** *auto*

**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*:  $\varphi \in \text{tf-mformula} \implies \varphi \in \mathcal{A} \implies \text{APR } \varphi \in \text{PLGl}$

**by** (*induct  $\varphi$  rule: tf-mformula.induct, auto intro!: sqcup sqcap*)

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

*ACC-cf-mf  $\varphi$*  = *ACC-cf (SET  $\varphi$ )*

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

*ACC-mf  $\varphi$*  = *ACC (SET  $\varphi$ )*

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

$\partial$ *Pos  $\varphi$*  = *POS*  $\cap$  *ACC-mf  $\varphi$*  - *ACC (APR  $\varphi$ )*

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

$\partial$ *Neg  $\varphi$*  = *ACC-cf (APR  $\varphi$ )* - *ACC-cf-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  $\psi$*

**unfolding**

*deviate-pos-def deviate-pos-cup-def*  
*deviate-neg-def deviate-neg-cup-def*  
*ACC-cf-mf-def ACC-cf-SET ACC-cf-union*  
*ACC-mf-def ACC-SET ACC-union*

**by** *auto*

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  $\psi$*

**unfolding**

*deviate-pos-def deviate-pos-cap-def*  
*ACC-mf-def ACC-SET ACC-odot*  
*deviate-neg-def deviate-neg-cap-def*  
*ACC-cf-mf-def ACC-cf-SET ACC-cf-odot*

**by** *auto*

**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* ( $\partial \sqcap Neg A B$ )  
**unfolding**  
*deviate-pos-def deviate-pos-cup-def deviate-pos-cap-def*  
*deviate-neg-def deviate-neg-cup-def deviate-neg-cap-def*  
**by** (*intro finite-subset*[*OF - finite-POS-NEG*], *auto*)<sup>+</sup>

Lemma 12

**lemma** *no-deviation*[*simp*]:  
 $\partial Pos FALSE = \{\}$   
 $\partial Neg FALSE = \{\}$   
 $\partial Pos (Var x) = \{\}$   
 $\partial Neg (Var x) = \{\}$   
**unfolding** *deviate-pos-def deviate-neg-def*  
**by** (*auto simp add: ACC-cf-mf-def ACC-mf-def*)

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*  $\varphi$ )  
**by** (*cases*  $\varphi$ , *auto intro: deviate-finite*)

**lemma** *finite-approx-neg*: *finite* (*approx-neg*  $\varphi$ )  
**by** (*cases*  $\varphi$ , *auto intro: deviate-finite*)

**lemma** *card-deviate-Pos*: **assumes** *phi*:  $\varphi \in tf\text{-mformula}$   $\varphi \in \mathcal{A}$   
**shows**  $card (\partial Pos \varphi) \leq cs \varphi * L^2 * ((m - l - 1) choose (k - l - 1))$

**proof** –

**let**  $?Pos = \lambda \varphi. \bigcup (approx\text{-pos} \text{ ‘ } SUB \varphi)$   
**have**  $\partial Pos \varphi \subseteq ?Pos \varphi$   
**using** *phi*

**proof** (*induct*  $\varphi$  *rule: tf-mformula.induct*)

**case** (*tf-Disj*  $\varphi \psi$ )

**from** *tf-Disj* **have** \*:  $\varphi \in tf\text{-mformula}$   $\psi \in tf\text{-mformula}$   $\varphi \in \mathcal{A}$   $\psi \in \mathcal{A}$  **by** *auto*

**note**  $IH = tf\text{-Disj}(2)[OF *(3)] tf\text{-Disj}(4)[OF *(4)]$

**have**  $\partial Pos (Disj \varphi \psi) \subseteq \partial \sqcup Pos (APR \varphi) (APR \psi) \cup \partial Pos \varphi \cup \partial Pos \psi$

**by** (*rule deviate-subset*)

**also have**  $\partial \sqcup Pos (APR \varphi) (APR \psi) = \{\}$

**by** (*rule deviate-pos-cup; intro APR \**)



**also have**  $\dots \cup \partial Pos \varphi \cup \partial Pos \psi \subseteq ?Pos \varphi \cup ?Pos \psi$  **using IH by auto**  
**also have**  $\dots \subseteq ?Pos (Disj \varphi \psi) \cup ?Pos (Disj \varphi \psi)$   
**by** (*intro Un-mono, auto*)  
**finally show**  $?case$  **by** *simp*  
**next**  
**case** (*tf-Conj*  $\varphi \psi$ )  
**from** *tf-Conj* **have**  $*$ :  $\varphi \in \mathcal{A} \psi \in \mathcal{A}$   
**by** (*auto intro: tf-mformula.intros*)  
**note**  $IH = tf-Conj(2)[OF *(1)] \text{ } tf-Conj(4)[OF *(2)]$   
**have**  $\partial Pos (Conj \varphi \psi) \subseteq \partial \Pi Pos (APR \varphi) (APR \psi) \cup \partial Pos \varphi \cup \partial Pos \psi$   
**by** (*rule deviate-subset*)  
**also have**  $\dots \subseteq \partial \Pi Pos (APR \varphi) (APR \psi) \cup ?Pos \varphi \cup ?Pos \psi$  **using IH by**  
*auto*  
**also have**  $\dots \subseteq ?Pos (Conj \varphi \psi) \cup ?Pos (Conj \varphi \psi) \cup ?Pos (Conj \varphi \psi)$   
**by** (*intro Un-mono, insert \*, auto*)  
**finally show**  $?case$  **by** *simp*  
**qed** *auto*  
**from** *card-mono[OF finite-UN-I[OF finite-SUB finite-approx-pos] this]*  
**have**  $card (\partial Pos \varphi) \leq card (\bigcup (approx-pos \text{ ' } SUB \varphi))$  **by** *simp*  
**also have**  $\dots \leq (\sum_{i \in SUB} \varphi. card (approx-pos i))$   
**by** (*rule card-UN-le[OF finite-SUB]*)  
**also have**  $\dots \leq (\sum_{i \in SUB} \varphi. L^2 * ((m - l - 1) choose (k - l - 1)))$   
**proof** (*rule sum-mono, goal-cases*)  
**case** (*1 psi*)  
**from** *phi 1* **have** *psi*:  $psi \in tf-mformula \text{ } psi \in \mathcal{A}$   
**by** (*induct*  $\varphi$  *rule: tf-mformula.induct, auto intro: tf-mformula.intros*)  
**show**  $?case$   
**proof** (*cases psi*)  
**case** (*Conj phi1 phi2*)  
**from** *psi this* **have**  $*$ :  $phi1 \in tf-mformula \text{ } phi1 \in \mathcal{A} \text{ } phi2 \in tf-mformula \text{ } phi2 \in \mathcal{A}$   
**by** (*cases rule: tf-mformula.cases, auto*)  
**from** *deviate-pos-cap[OF APR[OF \*(1-2)] APR[OF \*(3-4)]]*  
**show**  $?thesis$  **unfolding** *Conj* **by** (*simp add: ac-simps*)  
**qed** *auto*  
**qed**  
**also have**  $\dots = cs \varphi * L^2 * ((m - l - 1) choose (k - l - 1))$  **unfolding** *cs-def*  
**by** *simp*  
**finally show**  $card (\partial Pos \varphi) \leq cs \varphi * L^2 * ((m - l - 1) choose (k - l - 1))$  **by**  
*simp*  
**qed**  
  
**lemma** *card-deviate-Neg*: **assumes** *phi*:  $\varphi \in tf-mformula \text{ } \varphi \in \mathcal{A}$   
**shows**  $card (\partial Neg \varphi) \leq cs \varphi * L^2 * (k - 1)^m / 2^{(p - 1)}$   
**proof** –  
**let**  $?r = real$   
**let**  $?Neg = \lambda \varphi. \bigcup (approx-neg \text{ ' } SUB \varphi)$   
**have**  $\partial Neg \varphi \subseteq ?Neg \varphi$   
**using** *phi*

**proof** (*induct*  $\varphi$  *rule*: *tf-mformula.induct*)  
**case** (*tf-Disj*  $\varphi$   $\psi$ )  
**from** *tf-Disj* **have** \*:  $\varphi \in \text{tf-mformula}$   $\psi \in \text{tf-mformula}$   $\varphi \in \mathcal{A}$   $\psi \in \mathcal{A}$  **by** *auto*  
**note**  $IH = \text{tf-Disj}(2)[OF *(3)] \text{tf-Disj}(4)[OF *(4)]$   
**have**  $\partial Neg (\text{Disj } \varphi \psi) \subseteq \partial \sqcup Neg (APR \varphi) (APR \psi) \cup \partial Neg \varphi \cup \partial Neg \psi$   
**by** (*rule deviate-subset*)  
**also have**  $\dots \subseteq \partial \sqcup Neg (APR \varphi) (APR \psi) \cup ?Neg \varphi \cup ?Neg \psi$  **using**  $IH$  **by**  
*auto*  
**also have**  $\dots \subseteq ?Neg (\text{Disj } \varphi \psi) \cup ?Neg (\text{Disj } \varphi \psi) \cup ?Neg (\text{Disj } \varphi \psi)$   
**by** (*intro Un-mono, auto*)  
**finally show** *?case* **by** *simp*  
**next**  
**case** (*tf-Conj*  $\varphi$   $\psi$ )  
**from** *tf-Conj* **have** \*:  $\varphi \in \mathcal{A}$   $\psi \in \mathcal{A}$   
**by** (*auto intro: tf-mformula.intros*)  
**note**  $IH = \text{tf-Conj}(2)[OF *(1)] \text{tf-Conj}(4)[OF *(2)]$   
**have**  $\partial Neg (\text{Conj } \varphi \psi) \subseteq \partial \sqcap Neg (APR \varphi) (APR \psi) \cup \partial Neg \varphi \cup \partial Neg \psi$   
**by** (*rule deviate-subset*)  
**also have**  $\dots \subseteq \partial \sqcap Neg (APR \varphi) (APR \psi) \cup ?Neg \varphi \cup ?Neg \psi$  **using**  $IH$  **by**  
*auto*  
**also have**  $\dots \subseteq ?Neg (\text{Conj } \varphi \psi) \cup ?Neg (\text{Conj } \varphi \psi) \cup ?Neg (\text{Conj } \varphi \psi)$   
**by** (*intro Un-mono, auto*)  
**finally show** *?case* **by** *simp*  
**qed** *auto*  
**hence**  $\partial Neg \varphi \subseteq \bigcup (\text{approx-neg } 'SUB \varphi)$  **by** *auto*  
**from** *card-mono*[*OF finite-UN-I* [*OF finite-SUB finite-approx-neg*] *this*]  
**have**  $\text{card } (\partial Neg \varphi) \leq \text{card } (\bigcup (\text{approx-neg } 'SUB \varphi))$  .  
**also have**  $\dots \leq (\sum_{i \in SUB \varphi} \text{card } (\text{approx-neg } i))$   
**by** (*rule card-UN-le* [*OF finite-SUB*])  
**finally have**  $?r (\text{card } (\partial Neg \varphi)) \leq (\sum_{i \in SUB \varphi} \text{card } (\text{approx-neg } i))$  **by** *linarith*  
**also have**  $\dots = (\sum_{i \in SUB \varphi} ?r (\text{card } (\text{approx-neg } i)))$  **by** *simp*  
**also have**  $\dots \leq (\sum_{i \in SUB \varphi} L^2 * (k - 1)^m / 2^{(p - 1)})$   
**proof** (*rule sum-mono, goal-cases*)  
**case** (*1 psi*)  
**from** *phi 1* **have** *psi*:  $psi \in \text{tf-mformula}$   $psi \in \mathcal{A}$   
**by** (*induct*  $\varphi$  *rule*: *tf-mformula.induct, auto intro: tf-mformula.intros*)  
**show** *?case*  
**proof** (*cases psi*)  
**case** (*Conj phi1 phi2*)  
**from** *psi this* **have** \*:  $phi1 \in \text{tf-mformula}$   $phi1 \in \mathcal{A}$   $phi2 \in \text{tf-mformula}$   $phi2 \in \mathcal{A}$   
**by** (*cases rule: tf-mformula.cases, auto*) +  
**from** *deviate-neg-cap*[*OF APR*[*OF \*(1-2)*] *APR*[*OF \*(3-4)*]]  
**show** *?thesis unfolding Conj* **by** (*simp add: ac-simps*)  
**next**  
**case** (*Disj phi1 phi2*)  
**from** *psi this* **have** \*:  $phi1 \in \text{tf-mformula}$   $phi1 \in \mathcal{A}$   $phi2 \in \text{tf-mformula}$   $phi2 \in \mathcal{A}$   
**by** (*cases rule: tf-mformula.cases, auto*) +

```

from deviate-neg-cup[OF APR[OF *(1-2)] APR[OF *(3-4)]]
have card (approx-neg psi) ≤ ((L * 1) * (k - 1) ^ m) / 2 ^ (p - 1)
  unfolding Disj by (simp add: ac-simps)
also have ... ≤ ((L * L) * (k - 1) ^ m) / 2 ^ (p - 1)
  by (intro divide-right-mono, unfold of-nat-le-iff, intro mult-mono, insert L3,
auto)
  finally show ?thesis unfolding power2-eq-square by simp
qed auto
qed
also have ... = cs φ * L ^ 2 * (k - 1) ^ m / 2 ^ (p - 1) unfolding cs-def by simp
finally show card (∂Neg φ) ≤ cs φ * L ^ 2 * (k - 1) ^ m / 2 ^ (p - 1) .
qed

```

Lemma 12.3

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

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

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

**proof** –

**from** ne **obtain** E :: graph **where** Ephi:  $E \in \text{APR } \varphi$

**by** (auto simp: ACC-def accepts-def)

**from** APR[OF phi, unfolded PLG-def] Ephi

**have** EDl:  $E \in \mathcal{G}l$  **by** auto

**hence** vEl: card (v E) ≤ l **and** ED:  $E \in \mathcal{G}$

**unfolding** Gl-def Gl-def **by** auto

**have** E:  $E \in \mathcal{G}$  **using** ED[unfolded Gl-def] **by** auto

**have** sub:  $v E \subseteq [m]$  **by** (rule v-G[OF E])

**have** l ≤ card [m] **using** lm **by** auto

**from** exists-subset-between[OF vEl this sub finite-numbers]

**obtain** V **where** V:  $v E \subseteq V$   $V \subseteq [m]$  card V = l **by** auto

**from** finite-subset[OF V(2)] **have** finV: finite V **by** auto

**have** finPart: finite A **if**  $A \subseteq \{P. \text{partition-on } [n] P\}$  **for** n A

**by** (rule finite-subset[OF that finitely-many-partition-on], simp)

**have** finmv: finite ([m] - V) **using** finite-numbers[of m] **by** auto

**have** finK: finite [k - 1] **unfolding** numbers-def **by** auto

**define** F **where**  $F = \{f \in [m] \rightarrow_E [k - 1]. \text{inj-on } f V\}$

**have** FF:  $F \subseteq \mathcal{F}$  **unfolding** F-def F-def **by** auto

{

fix f

assume f:  $f \in F$

{

**from** this[unfolded F-def]

**have** f:  $f \in [m] \rightarrow_E [k - 1]$  **and** inj: inj-on f V **by** auto

**from** V l2 **have** 2: card V ≥ 2 **by** auto

**then obtain** x **where** x:  $x \in V$  **by** (cases V = {}, auto)

**have** card V = card (V - {x}) + 1 **using** x finV

**by** (metis One-nat-def add.right-neutral add-Suc-right card-Suc-Diff1)

**with** 2 **have** card (V - {x}) > 0 **by** auto

**hence** V - {x} ≠ {} **by** fastforce

**then obtain** y **where** y:  $y \in V$  **and** diff:  $x \neq y$  **by** auto

```

from inj diff x y have neq: f x ≠ f y by (auto simp: inj-on-def)
from x y diff V have {x, y} ∈ [m] 2 unfolding sameprod-altdef by auto
with neq have {x,y} ∈ C f unfolding C-def by auto
hence C f ≠ {} by auto
}
with NEG- $\mathcal{G}$  FF f have CfG: C f ∈  $\mathcal{G}$  C f ≠ {} by (auto simp: NEG-def)
have E ⊆ C f
proof
  fix e
  assume eE: e ∈ E
  with E[unfolded  $\mathcal{G}$ -def] have em: e ∈ [m] 2 by auto
  then obtain x y where e = {x,y} x ≠ y {x,y} ⊆ [m]
    and card: card e = 2
    unfolding binprod-def by auto
  from v-mem-sub[OF card eE]
  have {x,y} ⊆ v E using e by auto
  hence {x,y} ⊆ V using V by auto
  hence f x ≠ f y using e(2) f[unfolded F-def] by (auto simp: inj-on-def)
  thus e ∈ C f unfolding C-def using em e by auto
qed
with Ephi CfG have APR  $\varphi \Vdash$  C f
  unfolding accepts-def by auto
  hence f ∈ ACC-cf (APR  $\varphi$ ) using CfG f FF unfolding ACC-cf-def by auto
}
with FF have sub: F ⊆ ACC-cf (APR  $\varphi$ ) by auto
from card-mono[OF finite-subset[OF - finite-ACC] this]
have approx: card F ≤ card (ACC-cf (APR  $\varphi$ )) by auto
from card-inj-on-subset-funcset[OF finite-numbers finK V(2), unfolded card-numbers
V(3),
folded F-def]
have real (card F) = (real (k - 1)) ^ (m - l) * prod (λ i. real (k - 1 - i))
{0..<l}
  by simp
also have ... > (real (k - 1)) ^ m / 3
  by (rule approximation1)
finally have cardF: card F > (k - 1) ^ m / 3 by simp
with approx show ?thesis by simp
qed

```

Theorem 13

**lemma** theorem-13: **assumes** phi:  $\varphi \in$  tf-mformula  $\varphi \in \mathcal{A}$   
**and** sub: POS ⊆ ACC-mf  $\varphi$  ACC-cf-mf  $\varphi = \{\}$

**shows** cs  $\varphi > k$  powr (4 / 7 \* sqrt k)

**proof** –

**let** ?r = real :: nat ⇒ real

**have** cs  $\varphi > ((m - l) / k)^\lceil / (6 * L^2)$

**proof** (cases POS ∩ ACC (APR  $\varphi$ ) = {\})

**case** empty: True

**have**  $\partial$ Pos  $\varphi =$  POS ∩ ACC-mf  $\varphi -$  ACC (APR  $\varphi$ ) **unfolding** deviate-pos-def

**by** *auto*  
**also have**  $\dots = POS - ACC (APR \varphi)$  **using** *sub by blast*  
**also have**  $\dots = POS$  **using** *empty by auto*  
**finally have** *id*:  $\partial Pos \varphi = POS$  **by** *simp*  
**have** *m choose k = card POS* **by** (*simp add: card-POS*)  
**also have**  $\dots = card (\partial Pos \varphi)$  **unfolding** *id* **by** *simp*  
**also have**  $\dots \leq cs \varphi * L^2 * (m - l - 1 \text{ choose } (k - l - 1))$  **using**  
*card-deviate-Pos[OF phi]* **by** *auto*  
**finally have** *m choose k*  $\leq cs \varphi * L^2 * (m - l - 1 \text{ choose } (k - l - 1))$   
**by** *simp*  
**from** *approximation2[OF this]*  
**show**  $((m - l) / k)^l / (6 * L^2) < cs \varphi$  **by** *simp*  
**next**  
**case** *False*  
**have**  $POS \cap ACC (APR \varphi) \neq \{\}$  **by** *fact*  
**hence** *nempty*:  $APR \varphi \neq \{\}$  **by** *auto*  
**have**  $card (\partial Neg \varphi) = card (ACC\text{-}cf (APR \varphi) - ACC\text{-}cf\text{-}mf \varphi)$  **unfolding**  
*deviate-neg-def* **by** *auto*  
**also have**  $\dots = card (ACC\text{-}cf (APR \varphi))$  **using** *sub by auto*  
**also have**  $\dots > (k - 1)^m / 3$  **using** *ACC-cf-non-empty-approx[OF phi nempty]* .  
**finally have**  $(k - 1)^m / 3 < card (\partial Neg \varphi)$  .  
**also have**  $\dots \leq cs \varphi * L^2 * (k - 1)^m / 2^{p - 1}$   
**using** *card-deviate-Neg[OF phi]* **sub** **by** *auto*  
**finally have**  $(k - 1)^m / 3 < (cs \varphi * (L^2 * (k - 1)^m)) / 2^{p - 1}$  **by**  
*simp*  
**from** *approximation3[OF this]* **show** *?thesis* .  
**qed**  
**hence** *part1*:  $cs \varphi > ((m - l) / k)^l / (6 * L^2)$  .  
**from** *approximation4[OF this]* **show** *?thesis* **using** *k2* **by** *simp*  
**qed**

Definition 14

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

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

**lemma** *eval-gs-union*:

*eval-gs*  $\vartheta$   $(X \cup Y) = (\text{eval-gs } \vartheta X \vee \text{eval-gs } \vartheta Y)$   
**by** (*auto simp: eval-g-def*)

**lemma** *eval-gs-odot*: **assumes**  $X \subseteq \mathcal{G}$   $Y \subseteq \mathcal{G}$

**shows** *eval-gs*  $\vartheta$   $(X \odot Y) = (\text{eval-gs } \vartheta X \wedge \text{eval-gs } \vartheta Y)$

**proof**

```

assume eval-gs  $\vartheta$  ( $X \odot Y$ )
from this[unfolded eval-gs-def] obtain  $DE$  where  $DE: DE \in X \odot Y$ 
and eval: eval-g  $\vartheta$   $DE$  by auto
from  $DE$ [unfolded odot-def] obtain  $D E$  where  $id: DE = D \cup E$  and  $DE: D \in X E \in Y$ 
by auto
from eval have eval-g  $\vartheta$   $D$  eval-g  $\vartheta$   $E$  unfolding id eval-g-def
by auto
with  $DE$  show eval-gs  $\vartheta$   $X \wedge$  eval-gs  $\vartheta$   $Y$  unfolding eval-gs-def by auto
next
assume eval-gs  $\vartheta$   $X \wedge$  eval-gs  $\vartheta$   $Y$ 
then obtain  $D E$  where  $DE: D \in X E \in Y$  and eval: eval-g  $\vartheta$   $D$  eval-g  $\vartheta$   $E$ 
unfolding eval-gs-def by auto
from  $DE$  assms have  $D: D \in \mathcal{G} E \in \mathcal{G}$  by auto
let  $?U = D \cup E$ 
from eval have eval: eval-g  $\vartheta$   $?U$ 
unfolding eval-g-def by auto
from  $DE$  have  $1: ?U \in X \odot Y$  unfolding odot-def by auto
with  $1$  eval show eval-gs  $\vartheta$  ( $X \odot Y$ ) unfolding eval-gs-def by auto
qed

```

Lemma 15

```

lemma eval-set: assumes phi:  $\varphi \in \text{tf-mformula}$   $\varphi \in \mathcal{A}$ 
shows eval  $\vartheta$   $\varphi =$  eval-gs  $\vartheta$  (SET  $\varphi$ )
using phi
proof (induct  $\varphi$  rule: tf-mformula.induct)
case tf-False
then show  $?case$  unfolding eval-simps by simp
next
case (tf-Var  $x$ )
then show  $?case$  using inj-on- $\pi$  unfolding eval-simps
by (auto simp add: inj-on-def)
next
case (tf-Disj  $\varphi1$   $\varphi2$ )
thus  $?case$  by (auto simp: eval-gs-union)
next
case (tf-Conj  $\varphi1$   $\varphi2$ )
thus  $?case$  by (simp, intro eval-gs-odot[symmetric]; intro SET- $\mathcal{G}$ , auto)
qed

```

**definition**  $\vartheta_g :: \text{graph} \Rightarrow 'a \text{ VAS}$  **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 ( $\vartheta_g G$ )  $\varphi = (G \in \text{ACC-mf}$   $\varphi)$ 
using phi unfolding ACC-mf-def

```

```

proof (induct  $\varphi$  rule: tf-mformula.induct)
  case (tf-Var  $x$ )
  thus ?case by (auto simp: ACC-def  $G$  accepts-def  $\vartheta_g$ -def)
next
  case (tf-Disj  $\phi$   $\psi$ )
  thus ?case by (auto simp: ACC-union)
next
  case (tf-Conj  $\phi$   $\psi$ )
  thus ?case by (auto simp: ACC-odot)
qed simp

```

```

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-mformula
  and phi:  $\varphi \in \mathcal{A}$ 
  shows POS  $\subseteq$  ACC-mf  $\varphi$ 
proof
  fix  $G$ 
  assume POS:  $G \in$  POS
  with POS- $\mathcal{G}$  have  $G: G \in \mathcal{G}$  by auto
  with POS solution POS-CLIQUE
  have eval ( $\vartheta_g$   $G$ )  $\varphi$  by auto
  thus  $G \in$  ACC-mf  $\varphi$  unfolding eval-ACC[OF tf phi  $G$ ].
qed

```

```

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-mformula
  and phi:  $\varphi \in \mathcal{A}$ 
  shows ACC-cf-mf  $\varphi = \{\}$ 
proof (rule ccontr)
  assume  $\neg$  ?thesis
  from this[unfolded ACC-cf-mf-def ACC-cf-def]
  obtain  $F$  where  $F: F \in \mathcal{F}$  SET  $\varphi \Vdash C$   $F$  by auto
  define  $G$  where  $G = C$   $F$ 
  have NEG:  $G \in$  NEG unfolding NEG-def  $G$ -def using  $F$  by auto
  hence  $G \notin$  CLIQUE using CLIQUE-NEG by auto
  have GG:  $G \in \mathcal{G}$  unfolding  $G$ -def using  $F$ 
    using  $G$ -def NEG NEG- $\mathcal{G}$  by blast
  have GAcc: SET  $\varphi \Vdash G$  using  $F$ [folded  $G$ -def] by auto
  then obtain  $D ::$  graph where
     $D: D \in$  SET  $\varphi$  and sub:  $D \subseteq G$ 
    unfolding accepts-def by blast
  from SET- $\mathcal{G}$ [OF tf phi]  $D$ 
  have DG:  $D \in \mathcal{G}$  by auto
  have eval: eval ( $\vartheta_g$   $D$ )  $\varphi$  unfolding eval-set[OF tf phi] eval-gs-def
    by (intro bexI[OF -  $D$ ], unfold eval-g-def, insert DG, auto simp:  $\vartheta_g$ -def)
  hence  $D \in$  CLIQUE using solution[rule-format, OF DG] by auto
  hence  $G \in$  CLIQUE using GG sub unfolding CLIQUE-def by blast

```

**with**  $\langle G \notin \text{CLIQUE} \rangle$  **show** *False* **by** *auto*  
**qed**

## 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** –

**from** *CLIQUE-solution-imp-POS-sub-ACC*[*OF solution tf phi*] **have** *POS*:  $POS \subseteq \text{ACC-mf } \varphi$ .

**from** *CLIQUE-solution-imp-ACC-cf-empty*[*OF solution tf phi*] **have** *CF*:  $\text{ACC-cf-mf } \varphi = \{\}$ .

**from** *theorem-13*[*OF tf phi POS CF*]

**show** *?thesis* **by** *auto*

**qed**

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** –

**note** *vars* = *phi*[*unfolded A-def*]

**have** *CL*:  $\text{CLIQUE} = \text{Clique } [k^4] k \mathcal{G} = \text{Graphs } [k^4]$

**unfolding** *CLIQUE-def* *K-altdef* *m-def* *Clique-def* **by** *auto*

**with** *empty-CLIQUE* **have**  $\{\} \notin \text{Clique } [k^4] k$  **by** *simp*

**with** *solution*[*rule-format*, of  $\{\}$ ]

**have**  $\neg \text{eval } (\vartheta_g \{\}) \varphi$  **by** (*auto simp: Graphs-def*)

**from** *to-tf-mformula*[*OF this*]

**obtain**  $\psi$  **where**  $*$ :  $\psi \in \text{tf-mformula}$

( $\forall \vartheta. \text{eval } \vartheta \varphi = \text{eval } \vartheta \psi$ )  $\text{vars } \psi \subseteq \text{vars } \varphi$   $cs \psi \leq cs \varphi$  **by** *auto*

**with** *phi solution* **have** *psi*:  $\psi \in \mathcal{A}$

**and** *solution*:  $\forall G \in \mathcal{G}. (G \in \text{CLIQUE}) = \text{eval } (\vartheta_g G) \psi$  **unfolding** *A-def* **by** *auto*

**from** *Clique-not-solvable-by-small-tf-mformula*[*OF solution \*(1) psi*]

**show** *?thesis* **using**  $*(4)$  **by** *auto*

**qed**

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*:



```

  ∀ G ∈ Graphs [k^4]. G ∈ Clique [k^4] k ↔ eval (λ x. π x ∈ G) φ
  and vars: vars φ ⊆ V
shows cs φ > k powr (4 / 7 * sqrt k)
proof -
  {
    fix G
    assume G: G ∈ G
    have eval (λ x. π x ∈ G) φ = eval (ϑG G) φ using vars
      by (intro eval-vars, auto simp: ϑG-def)
  }
  have CL: CLIQUE = Clique [k^4] k G = Graphs [k^4]
    unfolding CLIQUE-def K-altdef m-def Clique-def by auto
  {
    fix G
    assume G: G ∈ G
    have eval (λ x. π x ∈ G) φ = eval (ϑG G) φ using vars
      by (intro eval-vars, auto simp: ϑG-def)
  }
  with phi-solves-clique CL have solves: ∀ G ∈ G. G ∈ CLIQUE ↔ eval (ϑG
G) φ
  by auto
  from vars have inA: φ ∈ A by (auto simp: A-def)
  from Clique-not-solvable-by-poly-mono[OF solves inA]
  show ?thesis by auto
qed
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: ∃ l. k = l^2
  and LARGE: k ≥ Large-Number
  and π: bij-betw π V [k^4]^2
  and solution: ∀ G ∈ Graphs [k^4]. (G ∈ Clique [k^4] k) = eval (λ x. π x ∈ G)
φ
  and vars: vars φ ⊆ V
  shows cs φ > k powr (4 / 7 * sqrt k)
proof -
  from k obtain l where kk: k = l^2 by auto
  note LARGE = LARGE[unfolded Large-Number-def]
  have k8: k ≥ 8^2 using LARGE by auto
  from this[unfolded kk power2-nat-le-eq-le]
  have l8: l ≥ 8 .
  define p where p = nat (ceiling (l * log 2 (k^4)))
  have tedious: l * log 2 (k^4) ≥ 0 using l8 k8 by auto
  have int p = ceiling (l * log 2 (k^4)) unfolding p-def

```

by (rule nat-0-le, insert tedious, auto)  
 from arg-cong[OF this, of real-of-int]  
 have rp: real p = ceiling (l \* log 2 (k ^ 4)) by simp  
 have one: real l \* log 2 (k ^ 4) ≤ p unfolding rp by simp  
 have two: p ≤ real l \* log 2 (k ^ 4) + 1 unfolding rp by simp  
 have real l < real l + 1 by simp  
 also have ... ≤ real l + real l using l8 by simp  
 also have ... = real l \* 2 by simp  
 also have ... = real l \* log 2 (2^2)  
 by (subst log-pow-cancel, auto)  
 also have ... ≤ real l \* log 2 (k ^ 4)  
 proof (intro mult-left-mono, subst log-le-cancel-iff)  
 have (4 :: real) ≤ 2^4 by simp  
 also have ... ≤ real k^4  
 by (rule power-mono, insert k8, auto)  
 finally show 2^2 ≤ real (k ^ 4) by simp  
 qed (insert k8, auto)  
 also have ... ≤ p by fact  
 finally have lp: l < p by auto  
 interpret second-assumptions l p k  
 proof (unfold-locales)  
 show 2 < l using l8 by auto  
 show 8 ≤ l by fact  
 show k = l^2 by fact  
 show l < p by fact  
 from LARGE have L0''^2 ≤ k by auto  
 from this[unfolded kk power2-nat-le-eq-le]  
 have L0''l: L0'' ≤ l .  
 have p ≤ real l \* log 2 (k ^ 4) + 1 by fact  
 also have ... < k unfolding kk  
 by (intro L0'' L0''l)  
 finally show p < k by simp  
 qed  
 interpret third-assumptions l p k  
 proof  
 show real l \* log 2 (real m) ≤ p using one unfolding m-def .  
 show p ≤ real l \* log 2 (real m) + 1 using two unfolding m-def .  
 from LARGE have L0^2 ≤ k by auto  
 from this[unfolded kk power2-nat-le-eq-le]  
 show L0 ≤ l .  
 from LARGE have L0'^2 ≤ k by auto  
 from this[unfolded kk power2-nat-le-eq-le]  
 show L0' ≤ l .  
 show M0' ≤ m using km LARGE by simp  
 show M0 ≤ m using km LARGE by simp  
 qed  
 interpret forth-assumptions l p k V π  
 by (standard, insert π m-def, auto simp: bij-betw-same-card[OF π])  
 from Clique-not-solvable-by-small-monotone-circuit-in-locale[OF solution vars]

**show** ?thesis .  
**qed**

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*:  $k \geq \text{Large-Number}$   
**and**  $\pi$ : *bij-betw*  $\pi$   $V$   $[k^{\wedge}8]^{\wedge}2$   
**and** *solution*:  $\forall G \in \text{Graphs } [k^{\wedge}8]. (G \in \text{Clique } [k^{\wedge}8] (k^{\wedge}2)) = \text{eval } (\lambda x. \pi x \in G) \varphi$   
**and** *vars*:  $\text{vars } \varphi \subseteq V$   
**shows**  $\text{cs } \varphi > k \text{ powr } (8 / 7 * k)$   
**proof** –

**from** *LARGE* **have** *LARGE*:  $\text{Large-Number} \leq k^2$   
**by** (*simp add: power2-nat-le-imp-le*)  
**have** *id*:  $k^2 \wedge 4 = k^{\wedge}8 \text{ sqrt } (k^{\wedge}2) = k$  **by** *auto*  
**from** *Clique-not-solvable-by-small-monotone-circuit-squared*[*of*  $k^{\wedge}2$ , *unfolded id, OF - LARGE*  $\pi$  *solution vars*]  
**have**  $\text{cs } \varphi > (k^{\wedge}2) \text{ powr } (4 / 7 * k)$  **by** *auto*  
**also** **have**  $(k^{\wedge}2) \text{ powr } (4 / 7 * k) = k \text{ powr } (8 / 7 * k)$   
**unfolding** *of-nat-power* **using** *powr-powr*[*of real*  $k$  2] **by** *simp*  
**finally** **show** ?thesis .

**qed**

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

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

**proof** –

**from** *kn* **obtain** *k* **where** *nk*:  $n = k^{\wedge}8$  **by** *auto*  
**have** *kn*:  $k = \text{root } 8 n$  **unfolding** *nk* *of-nat-power*  
**by** (*subst real-root-pos2, auto*)  
**have**  $\text{root } 4 n = \text{root } 4 ((\text{real } (k^{\wedge}2))^{\wedge}4)$  **unfolding** *nk* **by** *simp*  
**also** **have**  $\dots = k^{\wedge}2$  **by** (*simp add: real-root-pos-unique*)  
**finally** **have** *r4*:  $\text{root } 4 n = k^{\wedge}2$  **by** *simp*  
**have** *s*:  $s = k^{\wedge}2$  **using** *s* **unfolding** *r4* **by** *simp*  
**from** *large*[*unfolded nk large-number-def*] **have** *Large*:  $k \geq \text{Large-Number}$  **by** *simp*  
**have**  $0 < \text{Large-Number}$  **unfolding** *Large-Number-def* **by** *simp*

```

with Large have k0:  $k > 0$  by auto
hence n0:  $n > 0$  using nk by simp
from Clique-not-solvable-by-small-monotone-circuit[OF Large  $\pi$ [unfolded nk] -
vars]
  solution[unfolded s] nk
have real k powr (8 / 7 * real k) < cs  $\varphi$  by auto
also have real k powr (8 / 7 * real k) = root 8 n powr (8 / 7 * root 8 n)
  unfolding kn by simp
also have ... = ((root 8 n) powr (8 / 7)) powr (root 8 n)
  unfolding powr-powr by simp
also have (root 8 n) powr (8 / 7) = root 7 n using n0
  by (simp add: root-powr-inverse powr-powr)
finally show ?thesis .
qed

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.