UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 1. stopnja

Gregor Kikelj

PREVERJANJE KROMATSKEGA ŠTEVILA GRAFOV Z DOKAZOVALNIM POMOČNIKOM LEAN

Delo diplomskega seminarja

Mentor: prof. dr. Andrej Bauer

Ljubljana, 2024

Kazalo

1 Uvod **4**

2 Osnovne definicije **4**

3 Lean **5**

3.1 Računanje izrazov 5

3.2 Tipi 6

3.2.1 Strukture 7

4 Osnovne definicije v Leanu **7**

Preverjanje kromatskega števila grafov z dokazovalnim pomočnikom Lean

Povzetek

TODO

Verification of chromatic number of a graph with Lean proof assistant

Abstract

TODO

Math. Subj. Class. (2020): ..., ...
Ključne besede: ..., ...
Keywords: ..., ...

1 Uvod

Raziskovanje kromatskega števila grafov predstavlja pomemben problem teorije grafov. Kromatsko število grafa nam pove najmnaj koliko barv potrebujemo za barvanje vozlišč grafa, tako da dve sosednji vozlišči nista enake barve. Ta problem ni zanimiv le matematično, ampak ima uporabe tudi pri drugih področjih, predvsem v povezavi z računalništvom. Iskanje kromatskega števila je sicer računsko težek problem v smislu, da ne poznamo algoritma, ki bi ga računal v polinomskem času v odvisnosti od števila vozlišč grafa.

V tej diplomski nalogi se osredotočimo na iskanje kromatskega šteila skupaj z dokazom da smo res našli pravo vrednost. Da to dosežemo si bomo pomagali z dokazovalnikom Lean. V prvem delu naloge natančno definiramo pojme iz teorije grafov ki jih potrebujemo za dokazovanje. Nato na kratko opišemo osnove uporabe dokazovalnika Lean ter matematične definicije konstruiramo znotraj Leana. Na koncu definiramo nekaj algoritmov za iskanje kromatskega števila in dokažemo njihovo pravilnost.

2 Osnovne definicije

Definicija 2.1. Graf je urejen par $G = (V, E)$ kjer je

- $V \neq \emptyset$ končna množica vozlišč,
- E množica povezav, kjer je vsaka povezava množica dveh različnih vozlišč iz V.

Če sta različni vozlišči u, v elementa neke povezave iz E to pišemo kot $u \sim v$ ter pravimo, da sta sosednji vozlišči in da sta krajišči povezave $uv \in E$.

Definicija 2.2. Naj bo $v \in V$. Stopnja vozlišča v je število elementov E ki vsebujejo v, oziroma z besedami število povezav, ki vsebujejo v.

- Največjo stopnjo v grafu označimo z

$$\Delta(G) = \max_{v \in V} deg(v)$$

- Najmanjšo stopnjo v grafu označimo z

$$\delta(G) = \min_{v \in V} deg(v)$$

Definicija 2.3. Definiramo oznako $[k] = \{1, 2, \ldots, k-1\}$ za množico naravnih števil manjših od k.

Definicija 2.4. Naj bo G graf in $k \in \mathbb{N}$. Preslikava $f : V(G) \to [k]$ je k barvanje grafa G če za vse $u, v \in V$ velja $u \sim v \implies f(u) \neq f(v)$.

Z besedami: barvanje grafa s k barvami je preslikava, ki vsakemu vozlišču priredi eno izmed k barv, tako da sta sosednji vozlišči vedno pobarvani z različnima barvama.

Definicija 2.5. Kromatsko število grafa G ki ga označimo kot $\chi(G)$, je najmanjše število, da obstaja $\chi(G)$ barvanje grafa G.

Lema 2.6. *Naj bo G graf. Potem kromatsko število grafa G obstaja.*

Dokaz. Dovolj je da dokažemo da obstaja neko število k za katerega obstaja k barvanje grafa G. Dokažimo da je $k = |V|$ dovolj, kjer je V množica vozlišč grafa G. Ker je V končna množica lahko vozlišča oštevilčimo kot $V = \{v_0, v_1, \ldots, v_{k-1}\}$. Poglejmo si barvanje

$$f : V \to [k], f(v_i) = i$$

Dokazati moramo $\forall v_i, v_j \in V, v_i \sim v_j \implies f(v_i) \neq f(v_j)$. Po definiciji f je dovolj dokazati $\forall v_i, v_j \in V, v_i \sim v_j \implies i \neq j$ to pa je res ker sta si povezani vozlišči vedno različni po definiciji množice povezav grafa. □

Definicija 2.7. Naj bosta G in H grafa. Pravimo da je H podgraf G če je $V(H) \subset V(G)$ in $E(H) \subset E(G)$.

Lema 2.8. *Naj bo G graf in H njegov podgraf. Potem velja $\chi(H) \leq \chi(G)$.*

Dokaz. Očitno. □

3 Lean

Lean je interaktivni dokazovalnik. Uporabljali bomo verzijo Lean 4 ki je razvita pri Microsoftu in temelji na teoriji tipov s čimer se ne bomo ukvarjali preveč ker je tema te diplomske naloge teorija grafov. Lean 4 lahko in ga bomo uporabljali tudi kot funkcijski programski jezik.

3.1 Računanje izrazov

Kot ostali običajni programski jeziki tudi Lean podpira računanje osnovnih aritmetičnih izrazov.

```
#eval 1 + 1
```

Lean nam v informacijskem oknu pove da se izraz evaluira v 2. Lean upošteva tudi vrstni red operacij na primer

```
#eval 1 + 2 * 3
```

se izračuna v 7. Če želimo uporabiti drugačen vrstni red operacij lahko to naredimo z oklepaji

```
#eval (1 + 2) * 3
```

ki se izračuna v 9.

3.2 Tipi

Tipe poznamo iz drugih programskih jezikov (npr. Python, Java) so pa v Leanu bolj pomembni in imajo nekaj več funkcionalnosti.

Osnovni tipi s katerimi se bomo največ ukvarjali so:

- `Nat` naravna števila
- `Int` cela števila
- `String` nizi
- `List` seznami

Lean načeloma tipe izrazov ugotovi sam, lahko pa jih tudi podamo. Na primer

```
#eval 2
```

se izračuna v 2, Lean pa nam pove da je tip izraza `Nat`. Če pa želimo izraz izračunati kot `Int` lahko to naredimo z

```
#eval (2 : Int)
```

kar nam potem predstavlja 2 kot celo število. Lean ponavadi namesto `Nat` uporabi oznako $\mathbb{N}$ in namesto `Int` oznako $\mathbb{Z}$, to pa lahko uporabljamo tudi mi da se program bolj ujema z matematičnim zapisom.

Poglejmo si sedaj kako definiramo funkcije

```
def f (x : ℕ) : ℕ := x + 1
```

Definirali smo funkcijo `f` ki sprejme naravno število in vrne naravno število. Funkcijo lahko poračunamo podobno kot izraze

```
#eval f 2
```

ki se izračuna v 3. Če nismo prepričani kakšen tip ima nek objekt v leanu lahko to ugotovimo z ukazom

```
#check f
```

ki nam vrne $\mathbb{N} \to \mathbb{N}$ torej je funkcija f preslikava iz naravnih števil v naravna števila.

Lahko definiramo tudi funkcije več spremenljivk

```
def sum (x: ℕ) (y: ℕ) : ℕ := x + y
```

ki ima tip $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$ torej je preslikava iz naravnih števil v preslikave iz naravnih števil v naravna števila. Ker imata oba argumenta funkcije isti tip lahko definicijo skrajšamo

```
def sum (x y: ℕ) : ℕ := x + y
```

Ker lean zna sam ugotoviti tip vsote dveh naravnih števil tega ni potrebno podati je pa včasih to bolj berljivo

```
def sum (x y: ℕ) := x + y
```

Še krajše pa gre z lambda zapisom

```
λ x y:ℕ => x + y
```

3.2.1 Strukture

Podobno kot v večini programskih jezikov lahko tudi v Leanu definiramo strukture ki so ponavadi namenjene temu da združimo več podatkov v en objekt.

```
structure Tocka :=
    x: Nat
    y: Nat
```

Objekt neke strukture definiramo tako da navedemo vrednosti vseh polj

```
def izhodisce : Tocka := {x := 0, y := 0}
```

Tudi te tipe lahko uporabljamo kot argumente ali rezultate funkcij.

4 Osnovne definicije v Leanu

Pogjemo si sedaj kako smo definirali osnovne pojme iz teorije grafov v Leanu. Za berljivost ne bomo dodajali celotne kode ampak le odseke pomembne za razumevanje. Celotna koda je dostopna na https://github.com/grekiki2/Lean-tests.

```
structure Graph :=
    vertexSize : Nat
    connected: Fin vertexSize → Fin vertexSize → Pro
    connected_decidable: ∀ a b, Decidable (connected a b)
    irreflexive: ∀ n, ¬ connected n n
    symmetric: ∀ a b, connected a b → connected b a
```

Graf torej definiramo kot strukturo kjer podamo število vozlišč, izračunljivo funkcijo ki nam pove ali sta dve vozlišči povezani ter potrdilo da je relacija povezanosti irefleksivna in simetrična.

Za namene testiranja definiramo še graf K_n, ki je graf z n vozlišči in povezavami med vsemi pari vozlišč.

```
def K_n (k:Nat): Graph :=
  {vertexSize:=k, connected:=(λ x y=>x ≠ y), connected_decidable := by
    simp; intro a b;apply Not.decidable, irreflexive := by simp,
    symmetric:= by apply Ne.symm }
```

Na podoben način definiramo tudi C_n oziroma cikličen graf z n vozlišči.

Definiramo še tip ki predstavlja k barvanje grafa

```
def Coloring (G:Graph) (k : Nat) := Fin G.vertexSize → Fin k
```

Predstavili smo k barvanje grafa kot preslikavo iz (indeksa) vozlišča v eno izmed k barv. Definiramo še kdaj je k barvanje veljavno

```
def valid_coloring (G:Graph) {k:Nat} (coloring: Coloring G k): Prop :=
  ∀ a b, GraphConnected G a b → coloring a ≠ coloring b
```

Sledimo definiciji iz drugega poglavja. Torej z besedami, barvanje je veljavno če sta sosednji vozlišči vedno pobarvani z različnima barvama.

Ker preslikava preverja povezanost za končno število elementov se lahko uporablja za računanje. Drugače rečeno, smiselno je napisati

```
#eval valid_coloring (K_n 5) (λ x => 0)
```

in pričakovati da Lean to izračuna. Lean na žalost izračunljivosti tega izraza ne ugotovi sam, zato moramo pokazati, da je `valid_coloring` izračunjiv oz v Leanu decidable.

```
instance (G: Graph) (k:Nat) (coloring: Coloring G k): Decidable
    (@valid_coloring G k coloring)
    exact Nat.decidableForallFin _
```

Zdaj lahko torej za neko barvanje z Leanom preverimo ali je veljavno ali ne.

Slovar strokovnih izrazov

Chromatic number Kromatsko število