Minimizacija konačnih automata

Mooreov algoritam

Silva Haberl

PMF-Matematički odjel
Zagreb

rujan,2017.

Uvod

Konačni automat je uređena petorka $(Q, \Sigma, \delta, q_0, F)$,gdje:

Q konačan skup stanja

Σ ulazna abeceda (konačan neprazan skup simbola)

δ funkcija prijelaza $\delta : Q \times \Sigma \mapsto Q$

q_0 početno stanje automata

F skup završnih stanja

Minimizacija konačnog automata

Istovjetnost stanja

Stanje p automata M = $(Q, \Sigma, \delta, q_0, F)$ je istovjetno stanju p' ako i samo ako DKA M u stanju p prihvaća isti skup nizova kao i DKA M' = $(Q', \Sigma, \delta', q_0', F')$ u stanju p'.

Za bilokoji niz w skupa Σ^* mora vrijediti

uvjet podudarnosti:

$$\delta(p,w) \in F \quad i \quad \delta'(p',w) \in F'$$

ili

$$\delta(p,w) \notin F \quad i \quad \delta'(p',w) \notin F'$$

Istovjetnost DKA

DKA M i N su istovjetni ako i samo ako su istovjetna njihova početna stanja. Ispitivanje istovjetnosti stanja p i p' svodi se na ispitivanje dva uvjeta:

1.**Uvjet podudarnosti**

2.**Uvjet napredovanja**: Za bilo koji ulazni znak a mora vrijediti da su stanja $\delta(p,a) \quad i \quad \delta'(p',a)$ istovjetna.

Algoritam minimizacije konačnog automata

Neki od algoritama pronalaženja istovjetnih stanja, koji ispituju:

1.oba prethodna uvjeta

2.uvjeta podudarnosti (Mooreov algoritam)

3.pronalaženje neistovjetnih stanja

Mooreov algoritam dijeli skup stanja DKA M u podskupove na temelju uvjeta podudarnosti. Algoritam se izvodi u 3 koraka:

1. Skup stanja podijeli se u 2 grupe. U jednoj grupi su sva stanja koja su prihvatljiva (sva stanja $p \in F$), u drugoj grupi su sva stanja koja nisu prihvatljiva (sva stanja $p' \notin F$). Neka se znakom Π označi podjela skupa stanja u dvije grupe.

2. Primjeni se *Algoritam računanja nove podjele* na podjelu Π

3. Ako je podjela na grupe ostala ista, tj. ako je Π_{nova}= Π, stanja u istim

grupama su istovjetna. Ako je $\Pi_{nova} \neq \Pi$ onda se algoritam nastavlja od koraka (2) sa $\Pi = \Pi_{nova}$.

ALGORITAM RAČUNANJA NOVE PODJELE Π NOVA

za(sve grupe stanja G_j u podjeli Π)

$\{podijeligrupustanja$ G_j na podskupove tako da su

dva stanja p i p' iz iste grupe G_j u istom podskupu

ako i samo ako za svaki ulazni znak a vrijedi:

$\delta(p,a) \in G_i \delta(p',a) \in G_i$ gdje je G_i jedna od grupa stanja podjele Π;

// za različite ulazne znakove a grupe G_i mogu biti različite

označi novu podjelu skupa stanja Π_{nova}

U ovom seminaru implementaciju opisanog algoritma napravila sam u programskom jeziku Python. Opis korištenih metoda u implementaciji:

minDKA(automat)

Glavna metoda koja prima konačan automat, poziva ostale funkcije i vraća minimizirani konačni automat. Prvo se prolaskom po skupu stanja Q izvrši početna podjela na dvije klase po uvjetu podudarnosti. Prirodan broj je vrijednost pridjeljena ključu(stanju) u rječniku. Prirodan broj je oznaka klase u kojoj se nalazi svako stanje. U *while* petlji pozivamo metodu *NovaPodjela*. Ako je došlo do podjele, varijablu *jeLiBilaPodjela* postavljamo na vrijednost jedan, inače varijablu *podjelaGotova* postavljamo na jedan i *automatRadi* na nulu, te se automat zaustavlja. Ako je bila podjela, zapamtimo ju u varijabli i radimo novu podjelu, dok god možemo.

NovaPodjela(prethodnaPodjela,automat)

Metoda koja prima rječnik i automat te vraća rječnik nove podjele. Svaki rječnik se sastoji od ključeva i pripadne vrijednosti, odnosno od stanja i prirodnog broja koji označava pojedinu klasu kojoj to stanje pripada. Podjelu radi funkcija *podjeli* tako da joj pošaljemo trenutnu klasu stanja (ona koja za vrijednost imaju isti prirodan broj). Pozivamo ju onoliko puta koliko ima različitih klasa u tom trenu (podjeli).

podijeli(trenutnaKlasa,automat,brKlase,prethodnaPodjela,najveći)

Metoda prima jednu klasu iz trenutne podjele, sortira stanja unutar nje leksigografski te prvo stanje fiksira. Fiksno stanje uspoređuje sa ostalima u metodi *odredi*, idu li trenutna dva stanja na svaki znak iz alfabeta u istu klasu prethodne podjele po funkciji delta. Ako idu u istu, oba stanja ostaju u trenutnoj klasi inače , ovo koje nije fiksno bit će izbačeno u novo stvorenu klasu, samostalno. Metoda vraća rječnik, koji može sadržavati nove klase.

usporedi(stara,nova)

Metoda prihvaća dva rječnika, prethodne podjele i trenutne, uspoređuje stanja, tako da za svako stanje ispita je li se njegov broj klase promijenio.

Ako je, došlo je do nove podjele, te metoda vraća *True*, inače vraća *False*.

BrojRazličitihKlasa(tempNovi)

Metoda prima rječnik, prolazi po vrijednostima ključeva, prirodnim brojevima, i sprema ih u listu. Listu pretvara u skup. Skup sadrže sve elemente koji su međusobno različiti, tako da njegova duljina označava broj različitih klasa u trenutnoj podjeli (rječniku), tj. br.različitih prirodnih brojeva.

PromjenaNazivaStanja(rječnik, automat, najveći)

Metoda prihvaća zadnji generirani rječnik dobiven podjelama u podskupove, početni automat i najveći broj koji je upotrebljen u označavanju klasa. Da bi iz klasa koje sadrže istovjetna stanja, izbacili sve osim jednog, koristimo reverzni rječnik gdje su ključevi ,vrijednosti (brojevi klasa) zadnjeg generiranog rječnika, a vrijednosti reverznog su sada klase stanja .

U cilju smanjivanja broja stanja zadanog DKA, grupa istovjetnih stanja zamijeni se jedinstvenim stanjem sljedećim postupkom:

1. Najprije se iz grupa u kojima se nalaze istovjetna stanja (zadnje podjele u algoritmu) izbace sva stanja, osim jednog, koristeći listu stanja *lista* u kojoj su pohranjena stanja koja izbacujemo i varijablu

temp koja pamti jedno stanje koje ćemo zadržati iz te klase.

2. U skupu stanja Q ostavi se samo jedno od istovjetnih stanja iz grupe, a sva ostala istovjetna stanja se izbace. Isto tako i za skup F, završnih stanja. Koristimo python metodu *remove()*, jer su Q i F skupovi.

3. Sve oznake istovjetnih stanja u funkciji prijelaza delta, zamijene se oznakom onog stanja koje nije izbačeno iz grupe. Koristimo privremeni rječnik *d*.

4. Generiramo novi automat iz komponenti kojima je reduciran broj stanja, te ga vratimo glavnoj metodi *minDKA*.

Za Mooreov algoritam zaslužan je Edward F. More (1956.). Najgora vremenska složenost algoritma je $\Theta(n^2)$.