Minimizacija konačnog automata

Tihana Britvić

2. rujna 2017.

Sadržaj

Motivacija

Za proizvoljan DKA moguće je izgraditi beskonačno mnogo drugih DKA koji prihvaćaju isti jezik. Učinkovito programsko ostvarenje stoga zahtijeva izgradnju DKA sa što manjim brojem stanja, te vrijedi tvrdnja:

*Za regularni jezik **L** moguće je izgraditi DKA **M** koji ima manji ili jednak broj stanja od bilo kojeg drugog DKA **M'** koji prihvaća isti jezik.*

1 Uvod

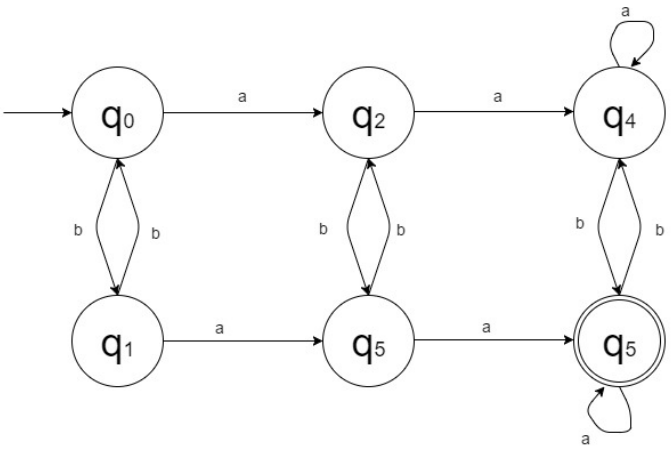
Za početak se prisjetimo nekih osnovnih pojmova. Uređenu petorku $(Q, \Sigma, \delta, q_0, F)$ zovemo determinističkim konačnim automatom (DKA) gdje je:

- Q konačan skup stanja,
- Σ konačan skup ulaznih znakova,
- δ funkcija prijelaza za koju vrijedi: $\delta : Q \times \Sigma \to Q$,
- $q_0 \in Q$ početno stanje,
- $F \subseteq Q$ skup završnih stanja.

Abecedu najčešće označavamo sa Σ, a njene elemente nazivamo znakovima. Riječ (nad abecedom Σ) označava niz od konačno mnogo (nula ili više) znakova iz Σ. Skup riječi nad istom abecedom zovemo jezik. Kažemo da automat M **prihvaća riječ** w $= \alpha_1\alpha_2...\alpha_n$ (nad abecedom od M) ako postoji niz stanja $r_0, r_1, ..., r_m$ takav da:

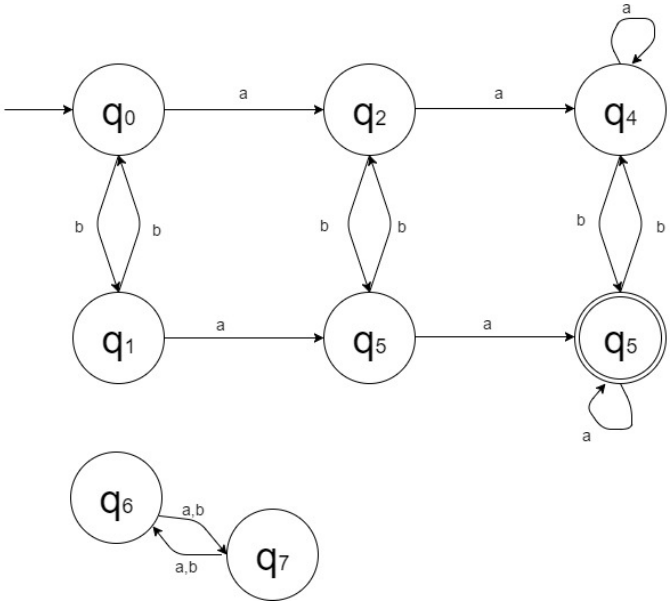
$$r_0 = q_0,\ \delta(r_{i-1}, \alpha_i) \ = \ r_i \ za\ i \in \{1, ..., m\}\ te\ r_m \in F. \tag{1}$$

Neka je M neki Kažemo da DKA M prepoznaje jezik L(M) := {w: M prihvaća w}, po definiciji (1).

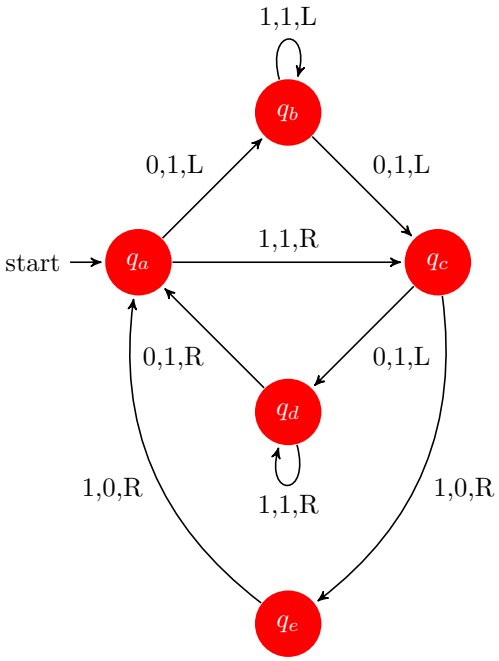


Slika 1: Dijagram stanja determinističkog konačnog automata koji prepoznaje jezik L = {w: w ima barem 2 znaka a i neparan broj znakova b} nad abecedom Σ = {a,b}

Prije nego definiramo pojam istovjentosti stanja i iskažemo algoritme za njih, naglasimo da postoji i drugi način označavanja DKA, a može se vidjeti na slici ispod.



Slika 2: Prošireni dijagram stanja deterministčkog konačnog automata koji prepoznaje jezik L = {w: w ima barem 2 znaka a i neparan broj znakova b} nad abecedom Σ = {a,b}. Primjetimo dodana su stanja q_6 i q_7.



Koristeći `sagetex` i okruženje `automata` možemo definirati jedan automat na sljedeći način:

```
A = Automaton([('P', 'Q', 0), ('P', 'P', 1),
               ('Q', 'P', 1), ('Q', 'Q', 0)],
              initial_states=['P'], final_states=['Q'])
```

Početno stanje je stanje P, a konačno stanje je označeno s Q. Ako DKA definiramo sa:

```
M = Automaton({'A': [('A', 0), ('B', 1), ('B', -1)], 'B': [('A', 0)]})
```

tada je zapravo zadana funkcija prijelaza δ, te s:

```
M.state('A').is_initial = True
```

provjeravamo je li A početno stanje, dok s:

```
M([1, -1])
sage: False
M([0, -1, 0, 1])
sage: True
```

vidimo da automat M prihvaća prvi ulaz, odnosno odbija drugi ulaz.

2 Istovjetnost stanja

Definicija 2.1. Neka je M neki DKA zadan s $M = (Q, \Sigma, \delta, q_0, F)$, te p stanje iz Q. Kažemo da je stanje *pistovjetno* stanju p' DKA $M' = (Q', \Sigma, \delta', q_0', F')$ ako i samo ako DKA M u stanju p prihvaća isti skup nizova kao i DKA M' u stanju p'. Za bilo koju riječ w skupa Σ^* vrijedi:

$$\delta(p, w) \in F \wedge \delta'(p', w) \in F' \tag{2}$$

ili

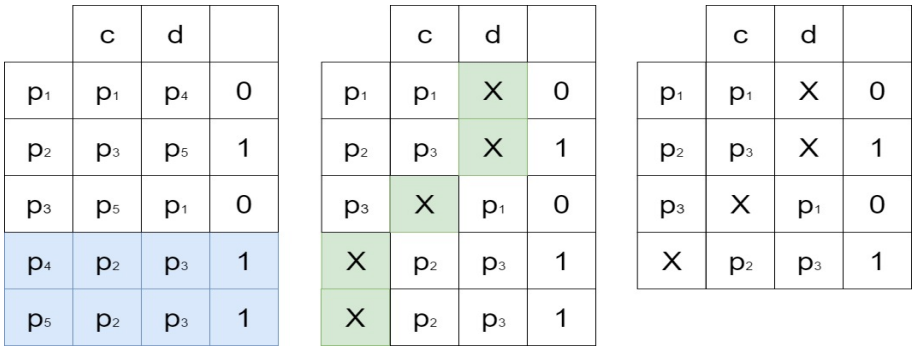
$$\delta(p, w) \notin F \wedge \delta'(p', w) \notin F \tag{3}$$

Napomena. Relacija istovjetnosti je tranzitivna relacija, tj. ako je stanje p istovjetno stanju q i stanje q istovjetno stanju r, tada je stanje p istovjetno stanju r. Svojstvo tranzitivnosti nam je važno zbog postupka redukcije broja stanja DKA.

Postupak redukcije može se opisati u tri koraka:

1. Sva istovjetna stanja označimo zajedničkim imenom izabranim u prethodnom koraku
2. Sve oznake istovjetnih stanja u funkciji prijelaza δ označe se izabranim zajedničkim imenom
3. U skupu Q ostavi se samo jedno od istovjetnih stanja, a sva ostala istovjetna stanja se izuzmu.

Na slici 3 možemo vidjeti postupak redukcije za istovjetna stanja p_4 i p_5.



Slika 3: Pojednsotavljenje DKA odbacivanjem jednog od istovjetnih stanja.

Prirodno nam se nameće pitanje: "Možemo li proširiti definiciju istovjetnosti stanja na istovjetnost DKA?" Odgovor je pozitivan i dan sljedećim teoremom.

Teorem 1. *Dva DKA M i M' su istovjetna ako i samo ako njihova početna stanja istovjetna.*

Ispitivanje istovjetnosti stanja p i q svodi se na ispitivanje dva uvjeta:

1. *Uvjet podudarnosti:* Stanja p i q moraju biti oba prihvatljiva ($p \in F \wedge q \in F$) ili neprihvatljia ($p \notin F \wedge q \notin F$)
2. *Uvjet napredovanja:* Za bilo koji ulazni znak a vrijedi da su stanja $\delta(p, a)$ i $\delta(q, a)$ istovjetna.

Korolar 1.1. *DKA je minimalan ako i samo ako vrijedi da je L(q) različito od L(q') za bilo koja dva različita stanja p i p'.*

Dokaz. $\Rightarrow$ Neka je A minimalan DKA. Svaki rezidual od L(A) je prepoznaje barem jedno od stanje od A. Kako je A minimalan, tada on ima brojna stanja, označimo ih s C_L, te vrijedi da je broj stanja jednak broju reziduala od L(A). Slijedi, broj različitih stanja od A prepoznaje različite reziduale od L(A).

$\Leftarrow$ Neka je A DKA takav da za različita stanja prepoznaje različite jezike. S obzirom da je svako stanje od A prepoznaje rezidual od L(A), i svaki rezidual od L(A) je prepoznatljiv nekim stanjem od A, broj stanja od jednak je broju reziduala od L(A). Slijedi, A ima jednako stanja kao i C_L, pa je minimalan.

□

3 Određivanje istovjetnih stanja

Sada ćemo obraditi osnovni algoritam za određivanje istovjetnih stanja. Te njegovu implementaciju u Pythonu.
Algoritam: Algoritam se zasniva na ispitivanju prethodno navedenih uvjeta podudarnosti i napredovanja koji slijede iz (2) i (3). Uvjeti se ispituju za sve parove stanja DKA.

```
Q, Σ, δ, q0, F = automat.komponente

def uvjet_podudarnosti(p,q):
    if (p in F and q in F):
        return True
    elif ((p not in F) and (q not in F)):
        return True
    else:
        return False
```

Slika 4: Implementacija uvjeta podudarnosti u Pythonu

```
def uvjet_napredovanja(p,q,a):
    if δ(p,a) == δ(q,a):
        return True
    else:
        return False
```

Slika 5: Implementacija uvjeta napredovanja u Pythonu za stanja p i q istog DKA

S obzirom da je potrebno ispitati sve parove stanja ovaj agoritam se smatra neučinkovitim te postoje poboljšanja. Tim algoritmima se nećemo baviti već se mogu pronaći u [1].
Prepostavimo da ispitujemo jesu li stanja p_0 i p_7 DKA datog na slici 3 istovjetna. Algoritam se izvodi u sljedećim koracima:

1. Tablica namijenjena ispitivanju istovjetnosti parova stanja gradi se tako da se za svaki ulazni znak igradi zasebni stupac. Izaberu se dva stanja za koja se želi ispitati istovjetnost, te se taj par upiše u prvi redak.

2. Uvjet podudarnosti ispituje se za sve parove novih stanja koji su zapisani u tablicu u prethodnom koraku. Ako par stanja nije podudaran, onda se odrede stanja za sve ulazne znakove, te se ti parovi upišu u odgovarajuće stupce tablice. Na primjer, za par stanja (p,q) i ulazni znak a, u stupac ulaznog znaka a upiše se novi par stanja $(\delta(p,a), \delta(q,a))$.

3. Ako su nova stanja u zadovoljila uvjet podudarnosti, novi par stanja u drugom koraku imamo tri mogućnosti dane u donjoj tablici.

```python
def iduci_korak(p, q, Σ, δ):
    redak = []
    for slovo in Σ:
        p_iduce = δ[(p, slovo)]
        q_iduce = δ[(q, slovo)]
        if p_iduce == q_iduce:
            continue
        par_iducih = (p_iduce, q_iduce)
        if par_iducih in redak:
            continue
        redak.append(par_iducih)
    return redak
```

Slika 6: Stvaranje novog retka u tablici implementirano je u Pythonu sa funkcijom `iduci_korak`. Funkcija za par stanja izvodi akciju upravo u ovisnosti o tablici ispod slike.

Par stanja	**Akcija**
Ista stanja	Nema akcije
Različita stanja za koje postoji zapis u jednom od prethodnih redaka	Nema akcije
Različita stanja za koje ne postoji zapis u jednom od prethodnih redaka	Stvori novi redak u tablici i upiši u njega novi par stanja

	c	d
p_0, p_7	p_0, p_6	p_3

Slika 7: Na slici možemo vidjeti primjer popunjavanja prvog retka u tablici kada vrijedi $\delta(p_0, c) = p_0$, $\delta(p_7, c) = p_6, \delta(p_0, d) = p_3, \delta(p_0, c) = p_0$

4. Ako se u trećem koraku ne zapiše niti jedan novi redak u tablici, onda je par stanja zapisan u prvom retku tablice istovjetan, kao i svi parovi stanja zapisani u ostalim recima tablice. Ostali parovi stanja su istovjetni jer vrijedi uvjet napredovanja.
   Ukoliko je novi redak zapisan u tablicu, algoritam nastavlja s drugim korakom.

```python
def istovjetna(p, q, Σ, δ, F):
    if not uvjet_podudarnosti(p, q, F):
        return False

    tablica_svih = [(p, q)]
    prethodni_korak = [(p, q)]
    while True:
        ubacili = False
        for (p_prethodni, q_prethodni) in prethodni_korak:
            novi_redak = iduci_korak(p_prethodni, q_prethodni, Σ, δ)
            for novi_par in novi_redak:
                if novi_par in tablica_svih:
                    continue
                elif not uvjet_podudarnosti(novi_par[0], novi_par[1], F):
                    return False
                else:
                    ubacili = True
                    tablica_svih.append(novi_par)
        if not ubacili:
            return True
```

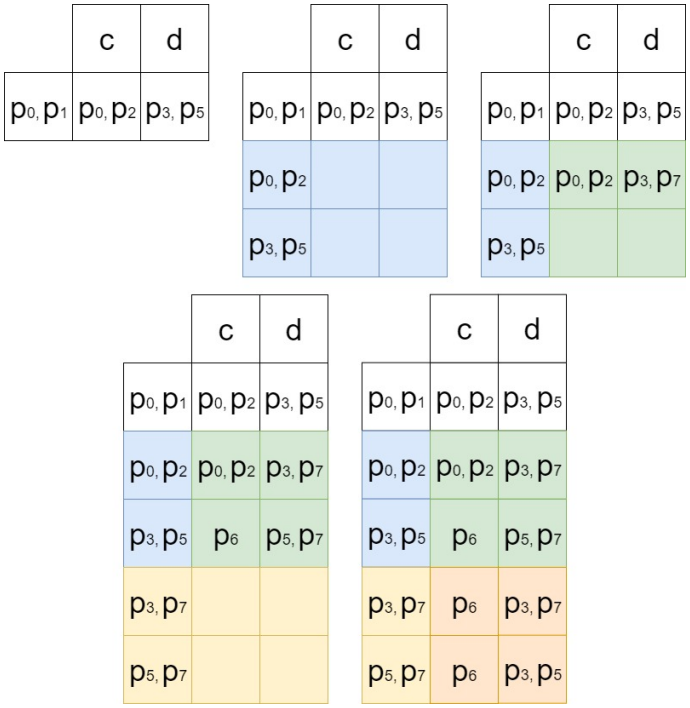
Slika 8: Funkcija `istovjetna` za nova dva stanja u tablici isputuje njihovu istovjetnost koristeći uvjet podudarnosti.

Na našem primjeru ispitivanja stanja p_0 i p_7 u drugom koraku drugog prolaza imamo ispitivanje podudarnost stanja p_0 i p_6. Budući da stanja p_0 i p_6 nisu podudarna, stanja p_0 i p_7 nisu istovjetna i algoritam se zaustavlja. U ovom slučaju funkcija `istovjetna` bi za stanja p_0 i p_7 vratila vrijednost `False` .
2.prolaz

	c	d
p_0, p_7	p_0, p_6	p_3

Slika 9: $(p_0 \notin F \wedge p_6 \in F) \Rightarrow stanja p_0$ i p_7 nisu istovjetna $\Rightarrow$ ALGORITAM SE ZAUSTAVLJA

Na slici je dokazana istovjetnost sljedećih parova stanja: (p_0, p_1), (p_0, p_2), (p_3, p_5), (p_3, p_7) i (p_5, p_7). Budući da za istovjetnost vrijedi tranzitivnost, na temelju istovjetnosti para stanja (p_0, p_1) i para stanja (p_0, p_2), par stanja (p_1, p_2) također je istovjetan. DKA prikazan na slici 2.10. pojednostavni se na sljedeći način: stanja p_0, $p_1 i\ p_2$ zamijene se stanjem A, a stanja p_3, $p_5\ i\ p_7$ zamijene se stanjem B. Pojednostavljani DKA prikazan je na slici 11 desno.



Slika 10: Prethodno opisan DKA čija su istovjetna stanja p_0, p_1, p_2 zamijenjena slovom A, a p_3, p_5, p_7 slovom B.

Implementacija minimizacije DKA dana je funkcijom `minimiziraj`. Zamijena istovjetih stanja radi uz pomoć fukcije `podijeli` koja jednostavno sva istovjetna stanja grupira u isti skup. Funkcija `minimiziraj` koristi tu informaciju kao rječnik te sva istovjetna stanja zamijeni s predstavnikom skupa (tj. s prvim stanjem skupa istovjetnih stanja).

	c	d	
p_0	p_0	p_3	0
p_1	p_2	p_5	0
p_2	p_2	p_7	0
p_3	p_6	p_7	0
p_4	p_1	p_6	1
p_5	p_6	p_5	0
p_6	p_6	p_3	1
p_7	p_6	p_3	0

	c	d	
A	A	B	0
B	p_6	B	0
p_4	A	p_6	1
p_6	p_6	B	1

Slika 11: Postupak dokazivanja istovjetnosti stanja p_0 i p_1 DKA datog na slici 10.

4 Nedohvatljiva stanja

Definicija 4.1. Za stanje p DKA $M = (Q, \Sigma, \delta, q_0, F)$ kažemo da je *nedohvatljivo* ako ne postoji niti jedan niz $w \in \Sigma^*$ za koji vrijedi da je $p = \delta(q_0, w)$. Stanje p_4 DKA sa slike 2.10 je primjer nedohvatljivog stanja. Odbacivanjem nedohvatljivih stanja dobije se istovjetni DKA manjim brojem stanja. Dohvatljiva stanja DKA $M = (Q, \Sigma, \delta, q_0, F)$ određuju se sljedećim algoritmom:

1. U listu dohvatljivih stanja DS upiše se početno stanje q_0.
2. Lista stanja DS proširi se skupom stanja $\{ \text{ p } | \text{ p } = \delta(q_0, a),\ za\ svea \in \Sigma\}$
3. Za sva stanja $q_i \in DS$ proširi se lista skupom stanja $\{p|p = \delta(q_0, a)$, stanje p nije prethodno zapisano u listu, za sve $a \in \Sigma\}$

Ako se lista dohvatljivih stanja DS proširi zapisom novog stanja, onda se rad algoritma nastavlja korakom (3). Lista DS sadrži sva dohvatljiva stanja, dok su ostala stanja nedohvatljiva.
Primjer rada algoritma.

1. korak DS: q_0
2. korak DS: q_0, q_1, q_5
   jer je $\delta(q_0, c) = q_1$, $\delta(q_0, d) = q_5$
3. korak DS: q_0, q_1, q_5, q_2, q_7, q_3
   jer je $\delta(q_1, c) = q_2$, $\delta(q_1, d) = q_7$ i $\delta(q_5, c) = q_3$
4. korak DS: q_0, q_1, q_5, q_2, q_7, q_3
   DS se ne mijenja, jer stanja q_2, q_7 i q_3 ne prelaze u niti jedno stanje koje od prije nije zapisano u listu, i rad algoritma se nastavlja.

Nedohvatljiva stanja q_4, q_6 i q_8 se odbacuju.

```
def minimiziraj(Q, δ, F, Σ, q0):
    Q1 = Q.copy()
    Σ1 = Σ.copy()
    δ1 = δ.copy()
    Q0 = q0
    F1 = F.copy()
    for stanja in podijeli(Q, δ, F, Σ, q0):
        zavrsno = False
        novo_stanje = stanja.pop()
        stanja.add(novo_stanje)
        for stanje in stanja:
            Q1.remove(stanje)
            for znak in Σ1:
                del δ1[(stanje, znak)]
            if stanje in F1:
                zavrsno = True
                F1.remove(stanje)
            Q1.add(novo_stanje)
            δ1[(novo_stanje, znak)] = δ[(novo_stanje,znak)]
            if zavrsno:
                F1.add(novo_stanje)

        novi_znak = stanja.pop()
        stanja.add(novi_znak)
        for stanje in stanja:
            for q in Q1:
                for znak in Σ:
                    if δ[(q, znak)] == stanje:
                            δ[(q, znak)] = novi_znak

    return KonačniAutomat.iz_komponenti(Q1, Σ1, δ1, Q0, F1)
```

Slika 12: Postupak dokazivanja istovjetnosti stanja p_0 i p_1 DKA datog na slici 10.

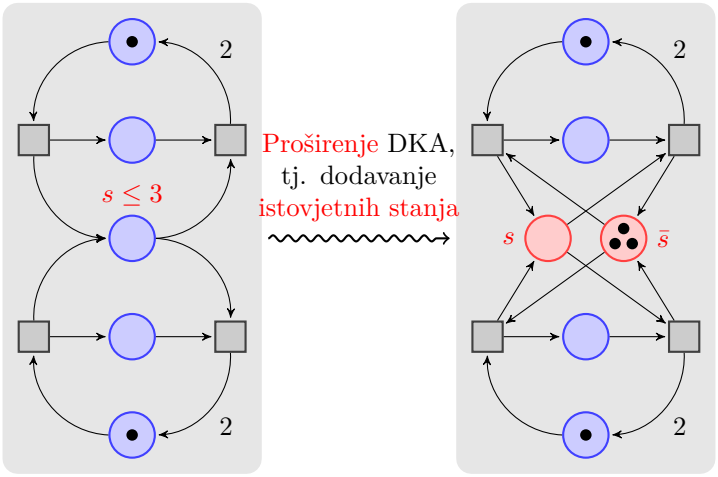
```
Out[31]: KonačniAutomat(abeceda=['c', 'd'], klasa=<class 'KA.KonačniAutomat'>, početno='p1', prijelaz={('p2', 'd'): 'p5', ('p5', 'c'):
          'p2', ('p5', 'd'): 'p3', ('p3', 'c'): 'p5', ('p1', 'c'): 'p1', ('p3', 'd'): 'p1', ('p1', 'd'): 'p5', ('p2', 'c'): 'p3'}, stanj
         a={'p5', 'p2', 'p3', 'p1'}, završna={'p5', 'p2'})
```

Slika 13: Za primjer iz [1], strana 28, slika 2.8. pokretanjem našeg programa dobijemo minimizirani DKA.

5 DKA s minimalnim brojem stanja

Odbacivanjem istovjetnih i nedohvatljivih stanja dobije se istovjetni DKA s minimalnim brojem stanja. Ne postoji niti jedan drugi DKA koji prihvaća drugi jezik, a ima manji broj stanja. Budući da je postupak odbacivanja nedohvatljivih stanja jednostavniji od postupka odbacivanja istovjetnih stanja, te budući da se odbacivanjem nedohvatljivih stanja smanjuje ukupni broj stanja, što pojednostavljuje postupak odbacivanja istovjetnih stanja, učinkovitije je najprije primijeniti postupak opdvacivanja nedohvatljivih stanja, a zatim postupak odbacivanja istovjetnih stanja.

Za kraj prikažimo još jednom proširenje DKA tako da mu dodamo još novo istovjetno stanje.



Literatura

[1] Siniša Srbljić, Jezični procesori 1, 2. izdanje, Zagreb, 2002.

[2] Sipser, Introduction to the theory of computation 2nd edition, Thomson, 2006.

[3] web.math.pmf.unizg.hr/nastava/ip/
Materijali s predavanja kolegija Interpretacija programa