Äärellisten automaattien ja säännöllisten lausekkeiden minimointi

Timi Suominen, Riia Ohtamaa ja Pessi Moilanen

Helsinki 3.2.2013
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Äärellisten automaattien minimointi

Minimaalisen automaatin määritelmä

Kaksi automaattia, jotka tunnistavat täsmälleen saman kielen ovat keskenään *ekvivalentteja.* Äärellinen automaatti on *minimaalinen,* jos se on tilamäärältään pienin ekvivalenttien automaattien joukossa. Automaatti, jossa on enemmän tiloja kuin ekvivalentissa minimaalisessa automaatissa on *redundantti.*

Miksi minimoida?

Automaatteja muodostavat algoritmit eivät aina tuota minimaalista automaattia. Minimaalisen automaatin käsittely on tehokkaampaa kuin *redundantin* ja on helpompi selvittää, tunnistavatko kaksi annettua automaattia saman kielen. Lisäksi ylimääräisiä tiloja on turha tallentaa.

Tunnettuja algoritmeja

Tehokkain algoritmi on Hopcroftin algoritmi 1970-luvulta, jonka me tulemme myöhemmin kattavasti esittelemään

Se näyttää wikipedian mukaan seuraavalta:

```
P := {F, Q \ F};
W := {F};
while (W is not empty) do
     choose and remove a set A from W
     for each c in ∑ do
          let X be the set of states for which a transition on c leads to a state in A
          for each set Y in P for which X ∩ Y is nonempty do
               replace Y in P by the two sets X ∩ Y and Y \ X
               if Y is in W
                    replace Y in W by the same two sets
               else
                    if |X ∩ Y| <= |Y \ X|
                         add X ∩ Y to W
                    else
                         add Y \ X to W
          end;
     end;
end;
```

Algoritmin rakenne:

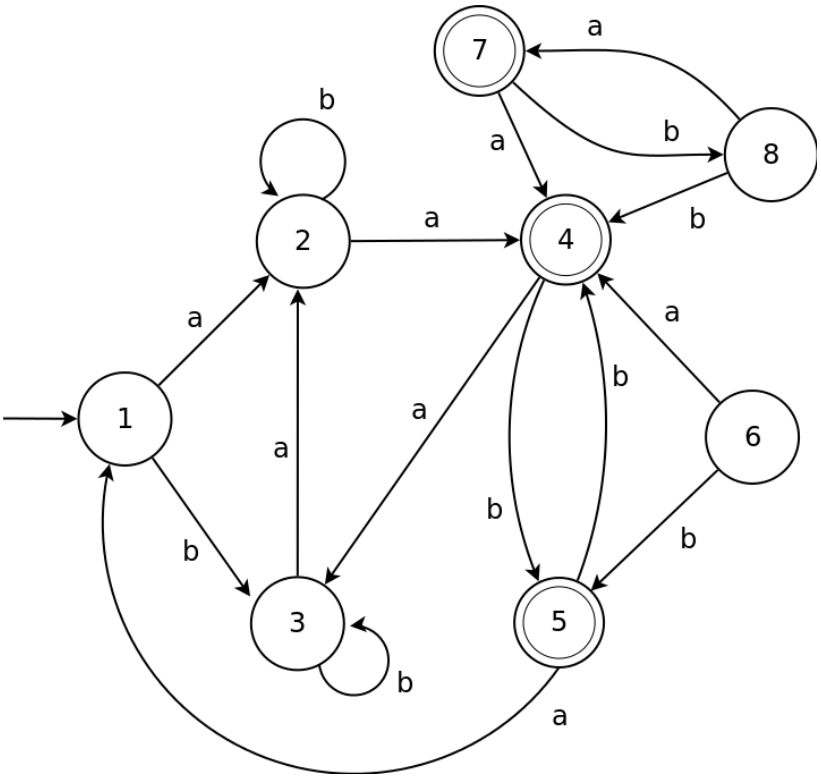
1. turhien tilojen poisto: poista M:stä kaikki tilat, joita ei voida saavuttaa tilasta q0 millään syötemerkkijonolla.

2. 0-ekvivalenssi: Osita M:n jäljelle jääneet tilat kahteen luokkaan: ei-lopputiloihin ja lopputiloihin.

3. k-ekvivalenssi -> (k + 1)-ekvivalenssi: Tarkasta, siirrytäänkö samaan ekvivalenssiluokkaan kuuluvista tiloista samoilla merkeillä aina samanluokkaisiin tiloihin. Jos kyllä, algoritmi päättyy ja minimiautomaatin M_{min} tilat vastaavat M:n tilojen luokkia.

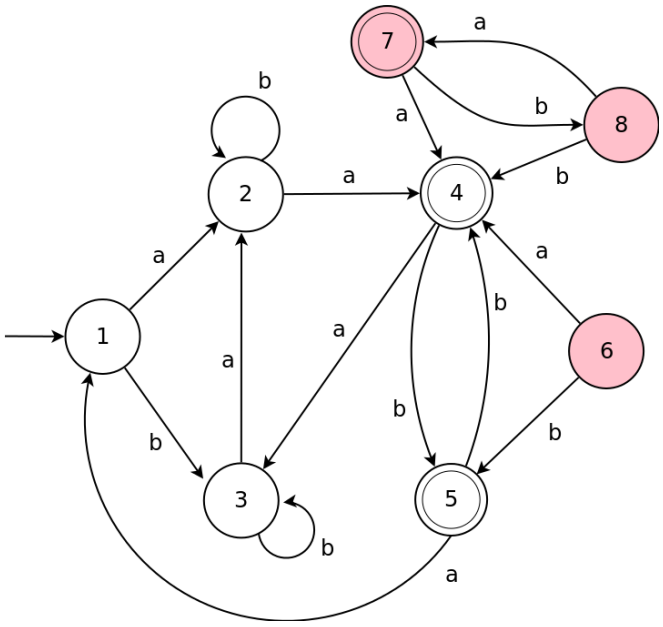
Esimerkki

Minimoidaan automaatti M:

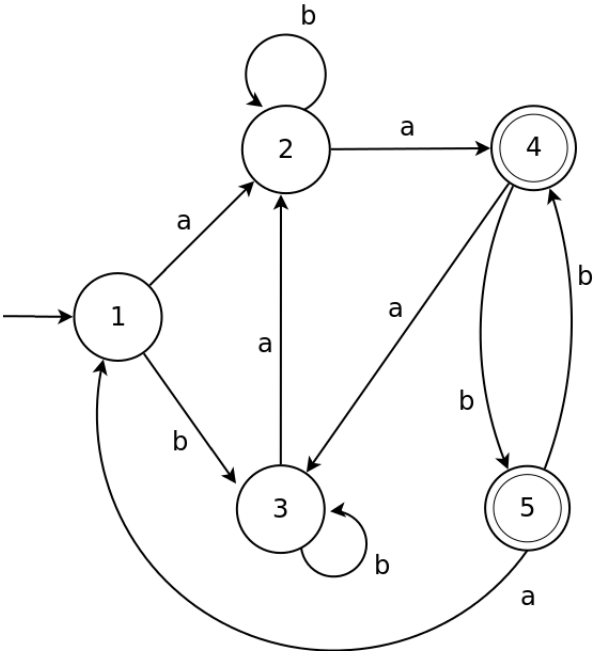


Vaihe 1:

Automaatin tiloihin 6, 7 ja 8 ei päästä alkutilasta millään syötteellä (kts. kuva 2), joten ne voidaan poistaa (kuva 3).



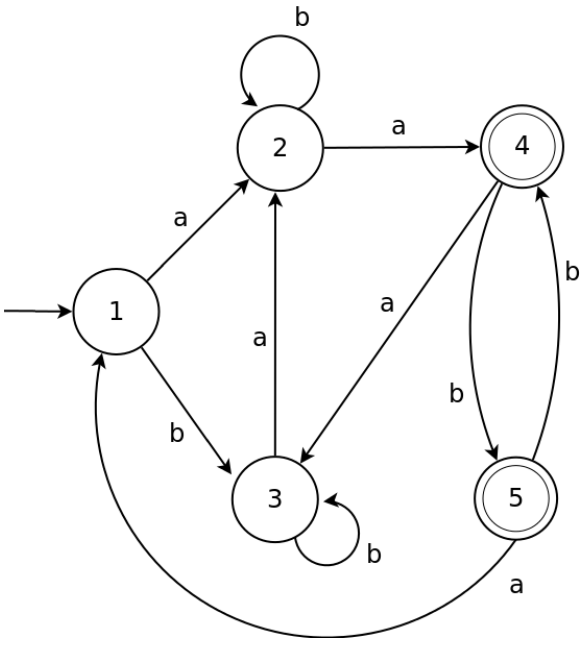
Kuva 2: Poistettavat tilat korostettu punaisella.



Kuva 3: Turhat tilat poistettu automaatista.

Vaihe 2:

Minimaalisen automaatin yksi tila esittää joukkoa DFA:n tiloja. Taulukoidaan siirtymät alkutilasta aakkoston kaikilla merkeillä. Nyt tilasta 1 pääsee merkillä a tilaan 2 ja b puolestaan vie tilaan 3. Tilasta 2 pääsee a:lla tilaan 4 ja b:llä tilaan 2. Jne.



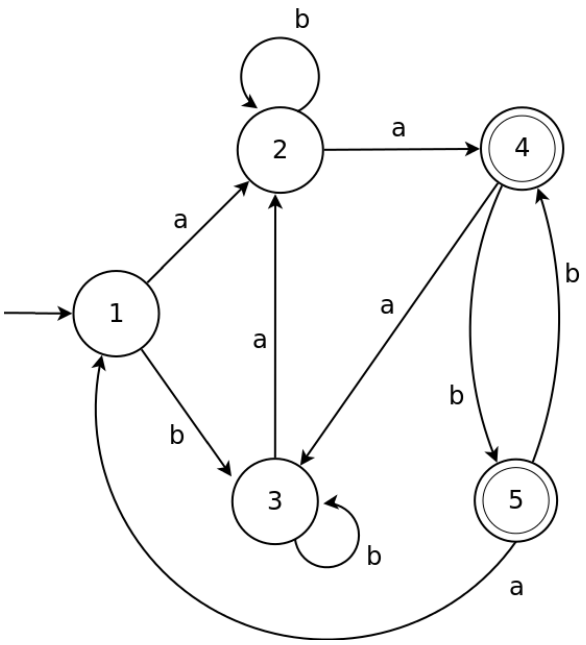
alkutila	a	b
1	2	3
2	4	2
3	2	3
4	3	5
5	1	4

Jaetaan jäljelle jääneet tilat lopputiloihin (ryhmä II) ja ei-lopputiloihin (ryhmä I)

alkutila	a	b
I: 1	2, I	3, I
2	4, II	2, I
3	2, I	3, I
II: 4	3, I	5, II
5	1, I	4, II

Vaihe 3:

Ekvivalenssiluokassa I on nyt kahden tyyppisiä tiloja ({1,3} ja {2}), joten ositusta täytyy hienontaa ja tarkastaa siirtymät uuden osituksen suhteen. Toisin sanoen ensimmäisessä luokassa a:lla pääsee sekä hyväksyvään että ei-hyväksyvään tilaan.



alkutila	a	b
I: 1 2 3	2, I 4, II 2, I	3, I 2, I 3, I
II: 4 5	3, I 1, I	5, II 4, II

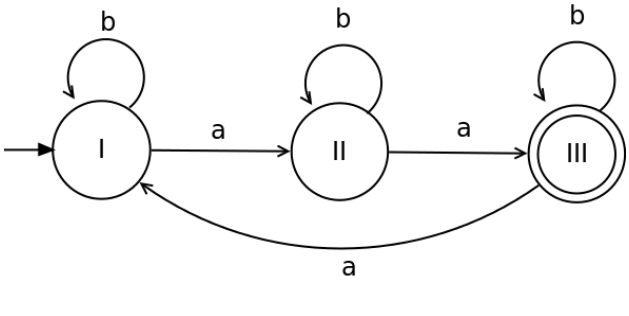
Ositetaan siirtymät uudestaan.

alkutila	a	b
I: 1 3	2, II 2, II	3, I 3, I
II: 2	4, III	2, II
III: 4 5	3, I 1, I	5, III 4, III

Huomataan, että esimerkissämme samaan ekvivalenssiluokkaan kuuluvista tiloista siirrytään *samoilla merkeillä* aina *samanluokkaisiin tiloihin*. Algoritmi päättyy ja minimiautomaatin M_{min} tilat vastaavat M:n tilojen luokkia.

Minimiautomaatin muodostus

Minimiautomaatin uusiksi tiloiksi tulevat vanhan automaatin ekvivalenssiluokat. Siirtymät löytyvät minimoinnissa syntyneestä taulukosta. Ekvivalenssitilasta I pääsee merkillä a tilaan II ja merkillä b tilaan I, tilasta II a:lla päästään tilaan III ja b:llä tilaan II, jne.



alkutila	a	b
I: 1 3	2, II 2, II	3, I 3, I
II: 2	4, III	2, II
III: 4 5	3, I 1, I	5, III 4, III

Ja näin minimiautomaatti on muodostettu.

NFA:n minimointi

[Sipser Thm. 1.39] Mille tahansa NFA:lle on olemassa DFA,
joka tunnistaa saman kielen.

NFA:n minimointialgoritmi

1. Muunna Sipserin lauseen 1.39 nojalla NFA DFA:ksi
2. Suorita esittelemämme minimointialgoritmi

Todistus

Todistetaan lopuksi, että esittelemämme algoritmi tuottaa minimaalisen automaatin.

Oletetaan, että on olemassa A:sta riippumaton automaatti N, joka hyväksyy saman kielen kuin A ja M, mutta jossa on vähemmän tiloja kuin automaatissa M. Voimme myös olettaa, että M:n tilat ovat eriävät.

Koska M ja N hyväksyvät saman kielen, ovat niiden alkutilat identtiset. Tiedetään, että jos tilat p ja q ovat identtiset, niin myös niiden vierustilat ovat. Näin ollen, koska M:n ja N:n alkutilat ovat identtiset, myös alkutilojen vierustilat ovat identtiset.

M ja N eivät sisällä turhia tiloja, koska muutoin löytyisi vähemmän tiloja sisältävä minimiautomaatti.
Tästä seuraa, että jokainen N:n tila on identtinen ainakin yhden M:n tilan kanssa.

Valitaan jokin M:n tila p ja N:n tila q. Syötteellä a1a2...ak päästään alkutilasta tilaan p ja samalla syötteellä alkutilasta myös tilaan q.

Aloitustilat ovat identtiset
=> aloitustiloja seuraavat tilat syötteellä a1 ovat identtiset ja näitä tiloja seuraavat tilat syötteellä a2 ovat identtiset jne.
=> tilat p ja q ovat identtiset.

N:ssä on vähemmän tiloja kuin M:ssä
=> M:ssä on kaksi tilaa p1 ja p2, jotka ovat identtiset jonkin N:n tilan q1 kanssa.
=> p1 ja p2 ovat identtiset.

Oletuksena oli etteivät M:n tilat ole identtisiä keskenään
=> RISTIRIITA
=> ei voi olla olemassa automaattia N joten M on A:n minimiautomaatti.

Säännöllisten lausekkeiden minimointi

Usein lähtökohtana lyhyen säännöllisen lausekkeen muodostamiselle on äärellinen automaatti. Silloin työjärjestys on seuraava: 1. automaatti minimoidaan, 2. muutetaan säännölliseksi lausekkeeksi ja 3. lauseke jälkikäsitellään. Varsinainen lausekkeen minimointi tapahtuu viimeisessä vaiheessa, jolloin sille etsitään lyhyempi esitysmuoto. Laaksosen mukaan tähän on olemassa algoritmeja, jotka etsivät suuren määrän minimaalisia lausekkeita sisältävästä tietokannasta lyhyitä vastineita lausekkeen osille.

Vaiheeseen 2 on olemassa muun muassa seuraavanlaisia tapoja

Tilanpoistomenetelmä

Kyseisessä menetelmässä tiloja aletaan harventaa ja niiden tilalle laitetaan säännöllisiä lausekkeita. Tämä menetelmä on kurssilta tuttu.

Reitinhakumenetelmä

Muuten samanlainen kuin tilanpoistomenetelmä, mutta tiloja ei poisteta laisinkaan. Floyd-Warshall-algoritmi muistuttaa kyseistä menetelmää.

Yhtälömenetelmä

Säännöllinen lauseke muodostetaan yhtälöryhmän avulla.

Vaiheessa 3 käytämme lyhennysalgoritmia

Lyhennysalgoritmi käyttää apunaan tietokantaa, joka sisältää lyhyempiä vastikkeita erilaisille säännöllisille lausekkeille. Algoritmi pyrkii korvaamaan lauseen osia tietokannasta löytyvillä lyhyemmillä vastineilla. Esimerkiksi (a | b | aa | bb)* vastaa lauseketta (a | b)*.

Algoritmin toimintatapa on Laaksosen mukaan seuraava:

1. Jos tietokannassa on lyhennettävän lausekkeen kuvaama kieli, palauta suoraan tietokannasta saatava lyhin lauseke.
2. Kerää listaan kaikki lyhennettävän lausekkeen osalausekkeet, joita voi muuttaa muusta lausekkeesta riippumattomasti.
3. Poista listasta saman lausekkeen toistuvat esiintymät.
4. Etsi listassa oleville osalausekkeille lyhyempiä vastineita tietokannasta.
5. Korvaa osalausekkeiden esiintymät lyhennettävässä lausekkeessa tietokannasta saaduilla lausekkeilla.

Lähteet

Antti Laaksonen, C-2010-51, Säännöllisten lausekkeiden minimointi
www.tcs.hut.fi/Studies/T-79.148/tekstit/luento04.pdf
cs.joensuu.fi/pages/whamalai/tepe04/tepe.pdf
Sipser, Introduction to the Theory of Computation, 2005