TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Joonas Liik 112814 IAPB

Unaarsete tsükliliste regulaarsete keelte pettehulkade eksperimentaalne uurimine

Bakalaureusetöö

Juhendaja: Hellis Tamm
PhD

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Joonas Liik

25.05.2021

Annotatsioon

Hiljuti pakkusid Tamm ja Merwe [9] välja piisava tingimuse, et unaarsel tsüklilisel keelel, millel on n vasakfaktorit, oleks pettehulk suurusega n. Lõin arvutiprogrammi, et eksperimentaalselt uurida, kas pettehulga olemasolust järeldub selle tingimuse täidetus. Leidsin näiteid, kus pettehulk suurusega n eksisteerib, kuid piisav tingimus sellise pettehulga eksisteerimiseks ei ole täidetud. Järelikult ei ole antud tingimus pettehulga suurusega n eksisteerimiseks tarvilik.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 22 leheküljel, 6 peatükki, 10 joonist, 1 tabelit.

Abstract

Experimental Study of Fooling Sets of Unary Cyclic Regular Languages

Recently a sufficient condition was proposed by Tamm and Merwe [9] for a regular unary cyclic language, with n left quotients, to have a fooling set of size n. I constructed a computer program to experimentally test if the existence of such a fooling set implies that the original constraints hold. I found that there exist cases where a fooling set of size n exists despite not fulfilling the sufficient condition. Thus, the sufficient condition for the existence of a fooling set of size n is not necessary.

The thesis is in Estonian and contains 22 pages of text, 6 chapters, 10 figures, 1 tables.

Lühendite ja mõistete sõnastik

DFA	*Deterministic Finite Automaton*, Deterministlik Lõplik Automaat
NFA	*Nondeterministic Finite Automaton*, Mittedeterministlik Lõplik Automaat
UCDFA	*Unary Cyclic Deterministic Finite Automaton*, Unaarne Tsükliline Deterministlik Lõplik Automaat
UCNFA	*Unary Cyclic Nondeterministic Finite Automaton*, Unaarne Tsükliline Mittedeterministlik Lõplik Automaat

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Automaatide teoorias on tähtsal kohal mittedeterministlike lõplike automaatide minimaalsusega seotud küsimused. Kuna regulaarse keele minimaalse mittedeterministliku automaadi leidmine on keerukas [8], siis on välja pakutud mõningaid alumise tõkke meetodeid mittedeterministliku automaadi olekute arvule, näiteks pettehulga meetodid (*fooling set techniques*) ([1], [5], [6]).

Käesolevas töös uurin unaarsete tsükliliste keelte (*unary cyclic languages*) pettehulki ja nende suurust. Selliste keelte minimaalseid mittedeterministlikke automaate on varem käsitletud [7]. Hiljuti esitasid Tamm ja Merwe [9] piisava tingimuse selleks, et pettehulga meetodiga saadav alamtõke oleks täpne unaarse tsüklilise keele jaoks, kuid selle tingimuse tarvilikkus pole tõestatud. Antud tingimus puudutab keele minimaalse deterministliku automaadi lõppolekute paiknemist automaadi struktuuris. Käesoleva töö eesmärk on aidata välja selgitada, kas mainitud tingimus on ainult piisav või ka tarvilik.

Uurimustöös esitan unaarsete tsükliliste keelte minimaalse deterministliku automaadi genereerimise algoritmi ja teostan selle implementatsiooni. Saadud arvutiprogrammi abil genereerin erinevaid unaarseid tsüklilisi automaate. Töötan välja meetodi, kuidas leida maksimaalse suurusega pettehulki selliste automaatide poolt aktsepteeritud keelte jaoks. Testin, kas pettehulga meetodiga saadud täpse alamtõkke puhul on ülalmainitud tingimus ([9], Lause 9) täidetud või mitte.

Töö tarvis valminud tarkvara [11] on kirjutatud programmerimiskeeles Python. Eraldi toon välja kasutatud algoritmid.

2 Lühiülevaade kasutatud kirjandusest

See töö toetub suures osas eelnevatele tulemustele.

Jiang, McDowell ja Ravikumar [7] uurivad mittedeterministlike automaatide erinevaid kujusid ja keerukust. Selles töös defineeritakse unaarsete tsükliliste automaatide jaoks „lihtne“ normaalkuju, mis on Chrobaki normaalkuju erijuht. Lisaks tõestatakse nende uuel kujul automaatide jaoks mitmed tähtsad põhiomadused.

Tamm ja Merwe [9] uurivad mittedeterministlike automaatide keerukusega seotud küsimusi, sõnastavad mitu automaatide keerukuse hindamise meetodit (pettehulkade meetod, laiendatud pettehulkade meetod ja kahealuselise graafi servade katte meetod bi-klikkidega) keele faktorite ja aatomite kaudu. Sealjuures pakuvad erilist huvi tulemused unaarsete tsükliliste keelte ja automaatide kohta. Selle töö tulemused on nii praeguse töö uurimisobjektiks ([9], Lause 9 pööratavus), kui ka annavad informatsiooni uuritavate automaatide struktuuri kohta. Samuti kasutan töös [9], Lause 10 tulemust, mille kohaselt on n vasakfaktoriga unaarsel tsüklilisel keelel, millel on pettehulk suurusega n, maksimaalselt $\lceil \frac{n}{2} \rceil$ lõppfaktorit. Sellele väitele tuginedes saan vältida automaatide genereerimist, millele vastav keel ei saa omada otsitava suurusega pettehulka.

3 Definitsioonid ja kasutatud tulemused

Selles peatükis annan ülevaate kasutatud mõistetest ja eelnevatest tulemustest, millele edaspidi tuginen.

3.1 Mittedeterministlik Lõplik Automaat (NFA)

Mittedeterministlik lõplik automaat (NFA) on viisik $N = (Q, \Sigma, \delta, I, F)$, kus Q on lõplik, mittetühi olekute hulk, Σ on lõplik mittetühi tähestik, $\delta : Q \times \Sigma \rightarrow 2^Q$ on siirdefunktsioon, $I \subseteq Q$ on algolekute hulk ja $F \subseteq Q$ on lõppolekute hulk. Laiendame siirdefunkstiooni funktsioonidega $\delta' : Q \times \Sigma^* \rightarrow 2^Q$ ja $\delta'' : 2^Q \times \Sigma^* \rightarrow 2^Q$ ning tähistame neid kõiki edaspidi δ.

3.2 Automaatide ekvivalentsus, paremkeel ja tühi olek

Regulaarne keel, mida aktsepteerib NFA N, on $L(N) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. Automaadi N oleku q paremkeeleks on $L_{q,F}(N) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$. Olek on tühi, kui tema paremkeel on tühi.

Olgu L regulaarne keel üle tähestiku Σ ja olgu A ja B automaadid, mis on defineeritud üle tähestiku Σ. Automaadid A ja B on ekvivalentsed, kui A ja B aktsepteerivad sama keelt.

3.3 Deterministlik Lõplik Automaat (DFA)

Deterministlik lõplik automaat (DFA) on viisik $D = (Q, \Sigma, \delta, q_0, F)$, kus Q on lõplik, mittetühi olekute hulk, Σ on lõplik mittetühi tähestik, $\delta : Q \times \Sigma \rightarrow Q$ on siirdefunktsioon, q_0 on algolek ja $F \subseteq Q$ on lõppolekute hulk. On hästi teada, et igal regulaarsel keelel on unikaalne minimaalne DFA.

3.4 Pöördautomaat ja pöördkeel

NFA $N = (Q, \Sigma, \delta, I, F)$ pöördautomaat on NFA $N^R = (Q, \Sigma, \delta^R, F, I)$, kus $q \in \delta^R(p, a)$ parajasti siis, kui $p \in \delta(q, a)$ ja $p, q \in Q$ ning $a \in \Sigma$.

Sõna $w = a_1a_2...a_n$ pöördsõnaks nimetatakse sõna $w^R = a_n...a_2a_1$. Keele L pöördkeeleks nimetatakse keelt L^R, mis koosneb kõigist keele L sõnade pöördsõnadest [10].

Kui automaat N aktsepteerib keelt L, siis N^R aktsepteerib keele L pöördkeelt L^R.

3.5 Regulaarse keele vasakfaktor

Regulaarse keele L vasakfaktor või lihtsalt faktor sõna $w \in \Sigma^*$ järgi on keel $w^{-1} L = \{x \in \Sigma^* \mid wx \in L\}$. Keele vasakfaktor on lõppfaktor, kui ta sisaldab tühja sõna ε. Keele L vasakfaktorid vastavad keele L minimaalse DFA olekutele. Keele L vasakfaktorid on keele L minimaalse DFA olekute paremkeeled.

3.6 Regulaarse keele aatomid

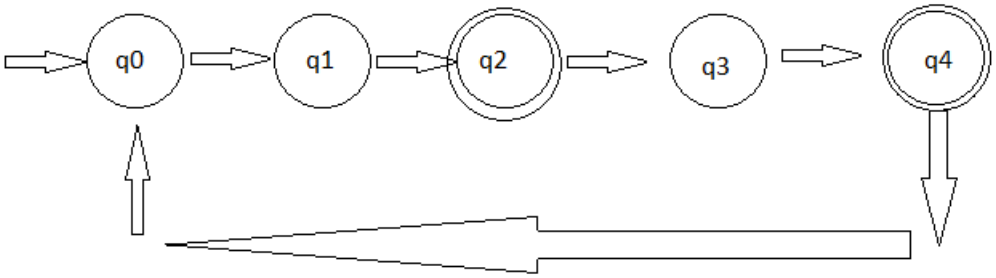
Olgu meil regulaarne keel L faktoritega $K_0, \ldots, K_{n-1}$. Siis selle keele aatomiks nimetame iga mittetühja keelt kujul $\tilde{K}_0 \cap \ldots \cap \tilde{K}_{n-1}$, kus $\tilde{K}_i$ on kas K_i või $\overline{K}_i$ *ja* $\overline{K}_i$ on K_i täiend Σ^* suhtes. Kui $\overline{K}_0 \cap \ldots \cap \overline{K}_{n-1}$ on aatom, siis seda nimetatakse *negatiivseks* aatomiks, kõiki teisi aatomeid nimetatakse *positiivseteks*. Keele L aatomid on vastavuses keele L pöördkeele L^R minimaalse DFA olekutega ja seega keele L^R faktoritega. Keele L aatomid on keele L pöördkeele L^R minimaalse DFA pöördautomaadi paremkeeled ([3],[4]).

3.7 Unaarne tsükliline automaat (UCDFA, UCNFA)

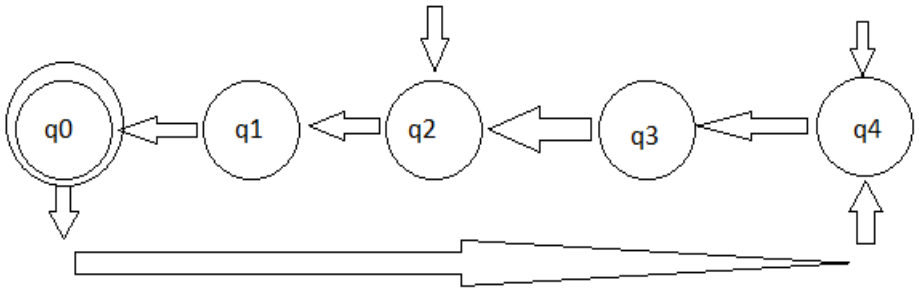
Olgu L regulaarne keel üle tähestiku $\Sigma = \{a\}$ ja olgu L minimaalne DFA $D = (Q, \Sigma, \delta, q0, F)$, mille olekute hulk on $Q = \{q_0, \ldots, q_{n-1}\}$ nii, et $\delta(q_i, a) = q_{i+1}$ kui $i = 0, \ldots, n-2$ ja $\delta(q_{n-1}, a) = q_0$ ning $F \subseteq Q$ on lõppolekute hulk. Sellist DFAd nimetatakse unaarseks tsükliliseks deterministlikuks automaadiks (UCDFA) ja keelt L nimetatakse unaarseks tsükliliseks keeleks ([9],[7]).

Olgu L regulaarne keel üle tähestiku $\Sigma = \{a\}$ ja olgu L NFA $N = (Q, \Sigma, \delta, I, F)$, mille olekute hulk on $Q = \{q_0, \ldots, q_{n-1}\}$ nii, et $\delta(q_i, a) = q_{i+1}$ kui $i = 0, \ldots, n-2$, ja $\delta(q_{n-1}, a) = q_0$, $I \subseteq Q$ on algolekute hulk ja $F \subseteq Q$ on lõppolekute hulk. Sellist NFAd nimetame unaarseks tsükliliseks mittedeterministlikuks automaadiks (UCNFA).

Näide: Joonisel 1 on kujutatud UCDFA $D = (Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{a\}, \delta(q_i, a) = q_{i+1 \pmod 5}, q_0, F=\{q_2, q_4\})$. Joonisel 2 on D pöördautomaat, UCNFA $N = (Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{a\}, \delta(q_i, a) = q_{i-1 \pmod 5}, I=\{q_2, q_4\}, F=\{q_0\})$.



Joonis 1. UCDFA algolekuga q_0 ja lõppolekutega q_2 ja q_4.



Joonis 2. Pöördautomaat, UCNFA algolekutega q_2 ja q_4 ning lõppolekuga q_0.

Unaarne keel L ja selle pöördkeel L^R on samad keeled [9]. Sellest tulenevalt on unaarse keele faktorid ja aatomid üks-ühele vastavuses [9].

3.8 UCDFA pöördautomaadi omadused

Olgu meil UCDFA, mille olekute hulk on Q, ja siirdefunktsioon $\delta(q_i, a) = q_{i+1 \bmod n}$. Kuna siirdefunktsioon δ defineerib üks-ühele vastavuse UCDFA olekute vahel, siis ka siirdefunktsiooni pöördfunktsioon $\delta^R(q_i, a) = q_{i-1 \bmod n}$ defineerib üks-ühele vastavuse UCDFA olekute vahel.

Teoreem 1. *UCDFA siirdefunktsiooni pöördfunktsioon defineerib üks-ühele vastavuse pöördautomaadi olekute vahel.*

3.9 Automaadi determiniseerimine

NFA N determiniseerimiseks nimetatakse DFA D leidmist, kus $D = N^D$ on ekvivalentne automaadiga N.

Selles töös kasutatud determiniseerimisalgoritm põhineb alamhulkade meetodil [10].

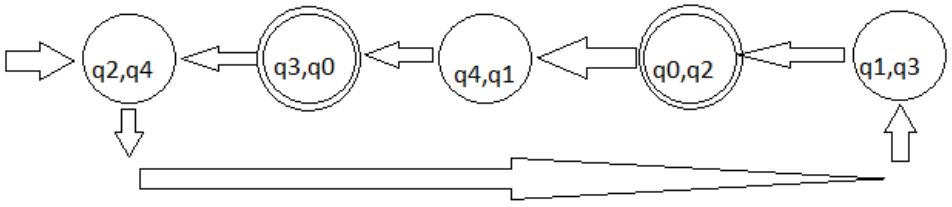
Olgu meil NFA $N = (Q_N, \Sigma, \delta_N, I_N, F_N)$. Olgu Q_D Q_N astmehulk.

Alustades automaadi algolekutest I_N, moodustame algse automaadi olekute alamhulkade $S \subseteq Q_D$ baasilt uue DFA olekud nii, et kõik olekud, mis on külastatavad mingist eelnevalt külastatud olekust ühe kindla sisendi peale, moodustavad hulga, mis on uue DFA olekuks. Kordame protsessi, kuni oleme kaardistanud kõik siirded.

Selles töös determiniseerin ainult UCNFAsid, mille sain UCDFA pööramise tulemusena ning kõikidel olekutel on üks sisenev ja üks väljuv siire. Determiniseeritavatel UCNFAdel on üks või rohkem algolekut, kuid täpselt üks lõppolek.

Tulenevalt UCNFA struktuurist, kui uue UCDFA olekute genereerimisel külastame varem nähtud Q_N alamhulka, siis teame, et oleme genereerinud kõik olekud.

Näide: Joonisel 3 on kujutatud Joonisel 2 esitatud UCNFA peale determiniseerimist.



Joonis 3. Pöördautomaat peale determiniseerimist, UCDFA.

3.10 Brzozowski teoreem

Järgmist tulemust tuntakse kui Brzozowski teoreemi [2]:

Teoreem 2. *Kui NFA N ei sisalda tühjasid olekuid ja N^R on deterministlik, siis N^D on minimaalne.*

Tulenevalt Teoreemist 2 on iga NFA *N* jaoks N^{RDRD} minimaalne DFA, mis on ekvivalentne *N*ga. Seda tulemust tuntakse kui Brzozowski DFA minimeerimise meetodit topeltpööramise abil.

Kuna unaarse keele ja tema pöördkeele minimaalne DFA on sama automaat, piisab unaarsete tsükliliste keelte DFAde minimeerimiseks Brzozowski topeltpööramise meetodiga ainult ühest pööramisest ja determiniseerimisest.

3.11 Keele vasakfaktorite ja aatomite maatriks.

Olgu meil regulaarne keel *L*, mille vasakfaktorite hulk on *K* ja positiivsete aatomite hulk *A*. Moodustame tabeli, nii et keele *L* mittetühjad vasakfaktorid on vastavuses tabeli ridadega ja keele *L* positiivsed aatomid on vastavuses tabeli tulpadega. Tabel sisaldab kohal (*i*, *j*) väärtust *1*, kui $A_j \subseteq K_i$, ja vastasel juhul *0*. Sellist tabelit nimetame keele *L* faktor-aatom maatriksiks ([9],[12],[13]).

Olgu meil minimaalne DFA *D*, mis aktsepteerib keelt *L* ja automaat D^{RD}, mille saame automaadi pööramise ja determiniseerimise kaudu. Faktor-aatom maatriksi saame luua *D* olekute, mis on vastavuses *D* vasakfaktoritega, ja D^{RD} olekute, mis vastavuses *D* aatomitega, põhjal. Maatriksi täitmiseks kontrollime iga *D* ja D^{RD} olekute paari $q_i \in D$ ja $q^{RD}{}_j \in D^{RD}$ kohta, kas $q^{RD}{}_j$ paremkeel on q_i paremkeele alamhulk. Kui $q_i \in q^{RD}{}_j$, siis on maatriksis *1*, muidu *0*.

3.12 Pettehulk ja laiendatud pettehulk

On kaks lihtsat meetodit NFA olekute minimaalse arvu hindamiseks: pettehulkade meetod (Glaister ja Shallit [5]) ja laiendatud pettehulkade meetod (Birget [1]).

Need kaks meetodit on esitatavad vastavalt järgneva teoreemi esimese ja teise juhuna:

Teoreem 3. *Olgu $L \subseteq \Sigma^*$ regulaarne keel ja oletame, et leidub paaride hulk*
$P = \{(x_i, w_i) \mid 0 \le i \le p-1\}$ *nii, et üks järgnevatest juhtudest kehtib:*
1. (a) $x_i w_i \in L$ *kui* $0 \le i \le p-1$,
(b) $x_j w_i \notin L$ *kui* $0 \le i, j \le p-1$, $i \ne j$,
või

2. (a) $x_i w_i \in L$ *kui* $0 \leq i \leq p - 1$,
(b) $x_j w_i \notin L$ *või* $x_i w_j \notin L$ *kui* $0 \leq i, j \leq p - 1$, $i \neq j$,
siis iga keelt L aktsepteerival NFAl on vähemalt p olekut.

Kui hulk *P* vastab Teoreemi 3 esimesele juhule, siis hulka *P* nimetatakse keele *L* pettehulgaks, kui teisele juhule, siis laiendatud pettehulgaks [6].

Tamm ja Merwe [9] defineerivad pettehulgad ja laiendatud pettehulgad läbi keele faktorite ja aatomite paaride:

Teoreem 4. *Olgu* $L \subseteq \Sigma^*$ *regulaarne keel ja oletame, et leidub faktorite-aatomite paaride hulk* $P = \{(K_i, A_i) \mid 0 \leq i \leq p - 1\}$ *nii, et üks järgnevatest juhtudest kehtib:*
1. (a) $A_i \subseteq K_i$ *kui* $0 \leq i \leq p - 1$,
(b) $A_i \not\subseteq K_j$ *kui* $0 \leq i, j \leq p - 1$, $i \neq j$,
või
2. (a) $A_i \subseteq K_i$ *kui* $0 \leq i \leq p - 1$,
(b) $A_i \not\subseteq K_j$ *või* $A_j \not\subseteq K_i$ *kui* $0 \leq i, j \leq p - 1$, $i \neq j$,
siis iga keelt L aktsepteerival NFAl on vähemalt p olekut.

Antud töös tegelen edaspidi ainult laiendatud pettehulkadega ja nimetan neid lihtsalt pettehulkadeks.

3.13 Piisav tingimus pettehulga olemasoluks

Olgu *L* unaarne tsükliline keel, millel on minimaalne UCDFA $D = (Q, \Sigma, \delta, q0, F)$, kus $Q=\{q_0, \ldots, q_{n-1}\}$.

Tamm ja Merwe [9] tõestavad UCDFAde kohta järgmise tulemuse:

Teoreem 5. *Kui automaadil D eksisteerib lõppolek* $q_i \in F$, *nii et iga* $q_j \in F$ *korral, kus* $j \neq i$, *olek* $q_{(2i-j) \bmod n}$ *ei ole lõppolek, siis keel L omab laiendatud pettehulka suurusega n.*

On lihtne näha, et kui DFA *D* rahuldab Teoreemi 5 piisavat tingimust ja vastavalt Teoreemile 5 keelel *L* on pettehulk suurusega *n*, siis selline pettehulk on keele *L* maksimaalne võimalik pettehulk, kuna keele *L* pettehulga suurus on keelt *L* aktsepteerivate NFAde ja seega DFAde olekute arvu alamtõkkeks. Kui keelel *L* oleks pettehulk suurusega $n + 1$, siis seda keelt ei saaks aktsepteerida DFA olekute arvuga *n*, seega pettehulk suurusega n on maksimaalne võimalik.

3.14 Unaarse tsüklilise keele pettehulk ja lõppfaktorite arv

Kasutame järgmist tulemust [9]:

Teoreem 6. *Kui unaarsel tsüklilisel keelel L on n vasakfaktorit ja pettehulk suurusega n, siis on keelel L maksimaalselt* $\lceil \frac{n}{2} \rceil$ *lõppfaktorit*.

4 UCDFAde ja pettehulkade genereerimine

Tamm ja Merwe [9] tuletavad ja tõestavad mitu huvitavat omadust tsükliliste keelte kohta. Selles töös uurin eksperimentaalselt Teoreemi 5 pööratavust.

Teoreemi 5 pööratavuse kontrollimiseks kirjutasin arvutiprogrammi, mis otsib minimaalseid UCDFAsid olekute arvuga *n*, mille puhul Teoreemi 5 piisav tingimus UCDFA poolt aktsepteeritava keele *L* pettehulga suurusega *n* olemasoluks ei ole täidetud ja keelel *L* eksisteerib pettehulk suurusega *n*.

On märkimisväärne, et UCDFAl on definitsioonist tulenevalt mitu huvitavat omadust, mis lihtsustavad rakendust: tulenevalt UCDFA siirdefunktsioonist $\delta(q_i, a) = q_{i+1 \pmod n}$, kui $n = |Q|$, piisab UCDFA kirjeldamiseks UCDFA olekute arvust *n* ja lõppolekute hulgast $F \subseteq Q$, kus $Q = \{q_i \mid 0 \leq i < n\}$, s.t. $UCDFA\ (Q, \Sigma, \delta, q_0, F) = (|Q|, F)$. Vastavust unaarsete tsükliliste keelte ja mittenegatiivsete täisarvude vahel on varem käsitlenud Jiang, McDowell ja Ravikumar [7]. See omadus lihtsustab automaatide genereerimist ja automaadi ja pettehulkade manipuleerimiseks kasutatud algoritmide implementatsiooni.

Olgu meil UCDFA olekute hulk *Q* ja olgu *PQ = P(Q)* hulga *Q* astmehulk. Siis kõikvõimalike UCDFAde suurusega |*Q*| hulk $P(D) = \{(|Q|, F_i) \mid F_i \subseteq PQ, kui\ |F_i|>0\}$. Sellest tulenevalt on kõikvõimalike automaatide genereerimine lihtne. Selles töös genereerin kõikvõimalikke huvi pakkuva suurusega automaate ja filtreerin siis välja huvi pakkuvad näited (minimaalsed DFAd).

Kirjutatud programm [11] ja koodi näited on programmeerimiskeeles Python.

4.1 Minimaalsete UCDFAde genereerimine

Minimaalsete UCDFAde genereerimise algoritm on väga naiivne: genereeritakse kõikvõimalikud UCDFAd suurusega $n = |Q|$, seejärel filtreeritakse genereeritud kandidaadid minimaalsuse alusel.

Kandidaatide filtreerimiseks konstrueerin parameetrite põhjal DFA ja kontrollin selle DFA minimaalsust.

```
from itertools import combinations

def _final_state_candidates(number_of_states):
    number_of_final_states = (number_of_states + 1) // 2
    for q in range(1, number_of_final_states + 1):
        yield from combinations(range(number_of_states), q)
```

Joonis 4. Genereeritava UCDFA lõppolekute hulkade kandidaatide genereerimine.

Kõikidel suurusega $n = |Q|$ UCDFAdel on sama olekute hulk Q, seega, nagu eelnevalt näidatud, erinevad suurusega n UCDFAd ainult lõppolekute hulga $F \subseteq Q$ poolest. Võttes arvesse seost pettehulga suuruse ja lõppolekute arvu vahel, saab piirata uuritavate minimaalsete UCDFAde lõppolekute hulka nii, et $|F| \leq \lceil \frac{|Q|}{2} \rceil$ (Teoreem 6).

Funktsioon _final_state_candidates leiab kõik Q alamhulgad suurusega vahemikus *1* kuni $\lceil \frac{n}{2} \rceil$. Need Q alamhulgad on huvitavate lõppolekute hulkade kandidaadid.

```
def generate_dfa_minimal(minQ: int, maxQ: int) -> Iterable[DFA]:
    for number_of_states in range(minQ, maxQ + 1):
        for F in _final_state_candidates(number_of_states):
            dfa = DFA(number_of_states, F)
            if dfa.is_minimal():
                yield dfa
```

Joonis 5. Minimaalsete UCDFAde genereerimine.

Minimaalsete automaatide genereerimiseks genereerin kõikvõimalikud lõppolekute hulgad otsitava suurusega automaadile, moodustan genereeritud lõppolekute hulkade põhjal UCDFA ja filtreerin leitud UCDFAd minimaalsuse põhjal.

```
def minimize(self) -> 'DFA':

    def get_previous_state(q: int) -> int:
        return q - 1 if q > 0 else originalN - 1

    def contains_original_starting_state(q):
        return 0 in q

    originalN = self.N
    cursor = frozenset(self.F)
    Q: List[FrozenSet[int]] = []

    while cursor not in Q:
        Q.append(cursor)
        cursor = frozenset(get_previous_state(q) for q in cursor)

    size_of_new_automata = len(Q)
    new_set_of_final_states = {
        index_of_new_final_state
        for index_of_new_final_state, q in enumerate(Q)
            if contains_original_starting_state(q)}

    return DFA(N=size_of_new_automata, F=new_set_of_final_states)
```

Joonis 6. UCDFA minimeerimine.

Minimeerimisalgoritm põhineb Teoreemil 2.

Minimeerimine käib läbi pöördautomaadi leidmise ja determiniseerimise. Unaarse tsüklilise DFA pöördautomaat on ekvivalentne algse automaadiga. Seega piisab vastavalt Teoreemile 2, kui pöörame ja determiniseerime automaadi ainult üks kord (üldjuhul 2 korda).

Pöördautomaadi olekute hulk on sama kui algse automaadi olekute hulk. Uueks algolekute hulgaks I' saab algse automaadi lõppolekute hulk F. Pöördautomaadi lõppolekuks on algse automaadi algolek q_0 ja siirdefunktsiooniks algse automaadi siirdefunktsiooni δ pöördfunktsioon δ^R.

UCNFA determiniseerimisel on kasulik funktsioon get_previous_state, mis UCDFA Q ja oleku q_i jaoks leiab oleku $q_{i-1 \ (mod \ |Q|)}$, ehk implementeerib UCDFA siirdefunktsiooni pöördfunktsiooni $\delta^R(q_j, a) = q_{i-1 \ (mod \ |Q|)}$.

Determiniseerin leitud pöördautomaadi alamhulkade meetodil (3.9). Tulenevalt UCDFA struktuurist on selge, et pöördautomaat on UCNFA (Teoreem 2). Tulenevalt sellest, et UCNFA olekud moodustavad ühe tsükli, piisab tarvilike pöördautomaadi olekute alamhulkade leidmiseks, kui alustan leitud pöördautomaadi algolekute hulgaga I' ja ülejäänud olekute alamhulgad leian *I′-st* alates, praeguse alamhulga kõikidele elementidele paralleelselt siirdefunktsiooni pöördfunktsiooni $\delta^R(q_j, a)$ korduvalt rakendades. Kui genereerin uuesti varem leitud olekute alamhulga, siis tean, et olen genereerinud kõik huvitavad alamhulgad. Uue UCDFA loomisel võtan algolekuks I' ja lõppolekuteks kõik genereeritud pöördautomaadi olekute alamhulgad, mis sisaldavad lõppolekut. Leitud UCDFA siirdefunktsiooni δ defineerin nii, et olekute asukohad uues automaadis vastavad järjekorrale, milles neid genereerisin ja moodustavad tsükli.

Nüüd, kus pöördautomaat on determiniseeritud, saab seda kirjeldada kujul UCDFA $D = (n, F)$, kus n on UCDFA olekute arv ja F on lõppolekute hulk väljendatuna kaugusena algolekust.

Algselt genereeritud UCDFA minimaalsuse kontrollimiseks minimeerin genereeritud automaadi läbi pööramise ja determiniseerimise ning kontrollin, kas genereeritud ja minimeeritud automaadi puhul on tegu sama automaadiga. Kuna kõikvõimalike UCDFAde, millel on *1* kuni *n* olekut, genereerimise käigus genereerin loomulikult ka kõik *1* kuni *n* olekuga minimaalsed automaadid, siis piisab nende automaatide uurimisest, mis juba olid minimaalsed.

4.2 Automaadi pettehulkade leidmine

Enne automaadi pettehulga leidmist tean juba, et automaat, millele pettehulka leiame, ja selle automaadi pööramisel ja determiniseerimisel saadav automaat on sama automaat, sest teistsugused automaadid filtreerisin eelnevalt välja (4.1).

Unaarse tsüklilise keele, millel on n vasakfaktorit, maksimaalse pettehulga leidmiseks moodustan keele faktorite ja aatomite paaridest hulga suurusega n nii, et moodustatud hulk rahuldab pettehulkade omadusi (Teoreem 4).

Otsin ainult maksimaalse võimaliku suurusega pettehulki, sest need on Teoreemi 5 pööratavuse kontrollimiseks kõige huvitavamad ja piisavad.

Olgu meil UCDFA $D = (Q, \Sigma, \delta, q_0, F)$, mis aktsepteerib keelt L. Keele L pettehulga leidmiseks alustan D olekute hulgaga Q, mis on ühtlasi vastavuses keele L faktorite hulgaga ja aatomite hulgaga. Otsin pettehulka, mille elementide arv on sama, kui uuritava automaadi olekute arv.

```
def gen_fs_max_len_only(n, final_states):
    left = tuple(range(n))
    # A helper lookup table. This helps to generate only promising pairings.
    candidate_offsets = {
        x: tuple((f - x + n) % n for f in final_states)
        for x in left
    }

    def generate_candidate_pairings(cursor=0, current_candidate=None):
        if current_candidate is None:
            current_candidate = []
        if cursor == n:
            if is_valid_extended_fs(left, current_candidate, final_states,n):
                yield tuple(current_candidate)
            return
        x = left[cursor]
        for offset in candidate_offsets[x]:
            if offset not in current_candidate:
                current_candidate.append(offset)
                yield from generate_candidate_pairings(cursor + 1,
                    current_candidate)
                current_candidate.pop()

    for right in generate_candidate_pairings():
        yield left, right
```

Joonis 7. UCDFA pettehulkade genereerimine.

Pettehulkade genereerimiseks paaritan Q elemendid, rahuldades pettehulkade definitsioonis (Teoreem 4) seatud piiranguid.

Funktsioon gen_fs_max_len_only genereerib UCDFA olekute arvu ja lõppolekute asukohtade põhjal maksimaalseid pettehulki. Kõigepealt võtan pettehulga paaride vasakuteks elementideks kõik keele vasakfaktorid (automaadi olekud), sest eeldan, et tegu on minimaalse UCDFAga ja tahan leida pettehulka, millel on sama suurus kui antud automaadil. Tulenevalt sellest, et automaat D, millele pettehulka otsin, ja D pööramisel ja determiniseerimisel saadav automaat D^{RD}, on sama automaat piisab keele $L(D)$ faktor-

aatom maatriksi moodustamiseks ainult automaadi *D* olekutest, mis on vastavuses nii keele *L* faktorite kui aatomitega. *D* olekute ja lõppolekute põhjal konstrueerin tabeli candidate_offsets, mis sisaldab infot ainult nende faktor-aatom paaride kohta, mille puhul faktor-aatom maatriksis on väärtus *1* ning järelikult on nende faktor-aatom paaride puhul täidetud pettehulga sama paari elementide vaheline seos (Teoreem 4,2(a)): $A_i \subseteq K_i$. Seejärel, kasutades funktsiooni generate_candidate_pairings ja kasutades faktor-aatom paare tabelist candidate_offsets, otsin sobivat keele faktor-aatom paaride hulka, mis rahuldab (Teoreem 4, 2(b)) pettehulga eri paaride vahelisi piiranguid. Tingimuse (Teoreem 4, 2(b)) kontrollimiseks pettehulga kandidaadi puhul kasutan funktsiooni is_valid_extended_fs.

```
def is_valid_extended_fs(left, right, final_states, n):
    for i in range(1, len(left)):
        for j in range(i):
            xiyj = (left[i] + right[j]) % n
            xjyi = (left[j] + right[i]) % n
            if xiyj in final_states and xjyi in final_states:
                return False

    return True
```

Joonis 8. Pettehulga kandidaatide testimine.

Genereeritavate pettehulkade kandidaatide seast korrektsete pettehulkade välja valimiseks kontrollin funktsiooniga *is_valid_extended_fs*, kas leitud kandidaat vastab lisaks pettehulga sama paari faktori ja aatomi vahelisele piirangule (Teoreem 4, 2(a)): $A_i \subseteq K_i$, ka eri paari faktori ja aatomi vahelistele piirangutele (Teoreem 4, 2(b)): $A_i \not\subseteq K_j$ *või* $A_j \not\subseteq K_i$ *kui* $i \neq j$. Funktsioon *is_valid_extended_fs* kontrollib ükshaaval kõikide eri paaride faktorite ja aatomite puhul, kas need rahuldavad tingimust (Teoreem 4, 2(b)) ning kui ühtegi tingimust mitterahuldavat paari ei leidu, siis on tegu korrektse pettehulgaga.

Näide: Joonisel 1 esitatud automaadi UCDFA $D = (Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{a\}, \delta(q_i, a) = q_{i+1 \pmod 5}, q_0, F=\{q_2, q_4\})$, mis täidab pettehulga suurusega 5 olemasoluks piisavat tingimust, jaoks leiab algoritm pettehulga *{(0, 2), (1, 1), (2, 0), (3, 4), (4, 3)}*.

5 Unaarse tsüklilise keele pettehulga piisav tingimus ja selle pööratavus

Realiseerisin algoritmi unaarse tsüklilise keele maksimaalse pettehulga olemasoluks piisava tingimuse (Teoreem 5) kontrollimiseks.

```
    def prop9(self) -> bool:
        n = self.N
        for i in self.F:
            # qj second state used to compute probing location
            for j in self.F:
                if i == j: continue
                probe = ((2 * i) - j + n) % n
                if probe in self.F:
                    break
            else:
                return True
        return False
```

Joonis 9. Funktsioon Teoreemi 5 maksimaalse pettehulga olemasoluks piisava tingimuse täidetuse kontrollimiseks.

Kuna UCDFA olekud on esitatud nende asukoha kaudu automaadi struktuuris, siis ei ole tarvis olekuid uuesti nummerdada ja algoritmi implementatsioon on märkimisväärselt otsene tõlge Teoreemi 5 definitsioonist.

```
def test_prop9_inverse(min_n=10, max_n=10):
    for n in range(min_n, max_n + 1):
        print(f"testing DFA's of size {n}: ", end="")
        for i, dfa in enumerate(generate_dfa_minimal(n, n)):
            if dfa.prop9():
                continue

            fs = dfa.get_fooling_set(extended=True,
                                     candidates=fs_candidates_max_len)

            if len(fs) == dfa.N:
                print()
                print(repr(dfa), "    ", str(dfa), "    ", fs, flush=True)

    print("--DONE--")
```

Joonis 10. Teoreemi 5 väite pöördkuju testimine.

Piisava tingimuse kontrollimiseks genereerisin erinevaid minimaalseid UCDFAsid. Neist valisin välja need, mille puhul piisav tingimus pettehulga suurusega $n = |Q|$ olemasoluks (Teoreem 5) ei olnud täidetud. Seejärel tõin esile need UCDFAd, millele vastaval keelel leidus pettehulk suurusega n.

Leidus mitmeid UCDFAsid, mille puhul piisav tingimus (Teoreem 5) ei olnud täidetud, kuid maksimaalne laiendatud pettehulk ikkagi leidus. Vähimad sellised näited on 10 olekuga automaadid, millest mõned näited on toodud järgnevas tabelis.

Tabel 1. Näited DFAdest ja nende maksimaalsetest pettehulkadest.

UCDFA	**Pettehulk**
DFA(Q={q0, q1, q2, q3, q4, q5, q6, q7, q8, q9}, Σ={a}, δ={0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 0}, q0, F={q0, q1, q2, q3, q5})	{(0,1), (1,4), (2,9), (3,2), (4,7), (5,0), (6,5), (7,8), (8,3), (9,6)}
DFA(Q={q0, q1, q2, q3, q4, q5, q6, q7, q8, q9}, Σ={a}, δ={0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 0}, q0, F={q0, q1, q2, q3, q8})	{(0,2), (1,7), (2,0), (3,5), (4,8), (5,3), (6,6), (7,1), (8,4), (9,9)}
DFA(Q={q0, q1, q2, q3, q4, q5, q6, q7, q8, q9}, Σ={a}, δ={0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6, 6: 7, 7: 8, 8: 9, 9: 0}, q0, F={q0, q1, q2, q4, q9})	{(0,0), (1,3), (2,8), (3,1), (4,6), (5,9), (6,4), (7,7), (8,2), (9,5)}

6 Kokkuvõte

Töö käigus realiseerisin arvutiprogrammi unaarsete tsükliliste deterministlike lõplike automaatide ja nendele vastavate unaarsete tsükliliste keelte pettehulkade genereerimiseks ja huvitavate omaduste uurimiseks. Programmi abil leidsin mitu juhtu, mille puhul maksimaalse pettehulga olemasoluks piisav tingimus (Teoreem 5) ei ole täidetud, kuid maksimaalne pettehulk siiski leidub. Järelikult ei ole antud tingimus tarvilik maksimaalse pettehulga eksisteerimiseks unaarse tsüklilise keele puhul.

Kasutatud kirjandus

[1] Birget, J.C.: Intersection and union of regular languages and state complexity. Inf. Process. Lett. 43, 185–190, 1992.

[2] Brzozowski, J. A. Canonical regular expressions and minimal state graphs for definite events. 1963 Proc. Sympos. Math. Theory of Automata (New York, 1962) pp. 529–561 Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.

[3] Brzozowski, J., Tamm, H.: Theory of ´atomata. In: Mauri, G., Leporati, A. (eds.)DLT 2011. LNCS, vol. 6795, pp. 105–116. Springer, Heidelberg, 2011.

[4] Brzozowski, J., Tamm, H.: Theory of ´atomata. Theor. Comput. Sci. 539, 13–27, 2014.

[5] Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inf. Process. Lett. 59, 75–77, 1996.

[6] Gruber, H., Holzer,M.: Finding lower bounds for nondeterministic state complexity is hard. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006, LNCS 4036, pp. 363–374. Springer, Heidelberg, 2006.

[7] Jiang, T., McDowell, E., Ravikumar, B.: The structure and complexity of minimal NFA's over a unary alphabet. Int. J. Found. Comput. Sci. 2(2), 163–182, 1991.

[8] Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM Journal on Computing, 22(6), 1117–1141, 1993.

[9] Tamm, H. ,van der Merwe, B.: Lower bound methods for the size of nondeterministic finite automata revisited. In: Drewes, F. et al. (eds.) LATA 2017, LNCS 10168, pp. 261-272. Springer, 2017.

[10] John E. Hopcroft, Jeffrey D. Ullman, Rajeev Motwani. Introduction to automata theory, languages, and computation, Edinburgh Gate : Pearson, 2014.

[11] J.Liik, https://bitbucket.org/Waffle_est/prop9

[12] Tamm, H.: New interpretation and generalization of the Kameda-Weiner method. In: 43rd International Colloquium on Automata, Languages, and Program-ming (ICALP 2016). Leibniz International Proceedings in Informatics (LIPIcs),vol. 55, pp. 116:1–116:12. Schloss Dagstuhl-Leibniz-Zentrum f̈ur Informatik, Dagstuhl (2016).

[13] Kameda, T., Weiner, P.: On the state minimization of nondeterministic finite automata. IEEE Trans. Comput.C–19(7), 617–627 (1970).

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Joonas Liik

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Unaarsete tsükliliste regulaarsete keelte pettehulkade eksperimentaalne uurimine“, mille juhendaja on PhD Hellis Tamm.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

25.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.