UNIVERZITET U NIŠU
PRIRODNO-MATEMATIČKI FAKULTET

Stefan P. Stanimirović

POBOLJŠANI ALGORITMI ZA DETERMINIZACIJU FAZI I TEŽINSKIH AUTOMATA

DOKTORSKA DISERTACIJA

Niš, 2019.

UNIVERSITY OF NIŠ
FACULTY OF SCIENCES AND MATHEMATICS

Stefan P. Stanimirović

IMPROVED ALGORITHMS FOR DETERMINIZATION OF FUZZY AND WEIGHTED AUTOMATA

DOCTORAL DISSERTATION

Niš, 2019.

Подаци о докторској дисертацији

Ментор:	Проф. Др Мирослав Д. Ћирић, редовни професор, Универзитет у Нишу, Природно-математички факултет
Наслов:	Побољшани алгоритми за детерминизацију фази и тежинских аутомата
Резиме:	Детерминизациони алгоритми представљају методе које за улазни фази (тежински) аутомат израчунавају језички еквивалентан комплетан детерминистички фази (тежински) аутомат, а нашли су примене у бројним областима, као што су лексикографска анализа, анализа регуларних израза, аутоматско препознавању говора, препознавању узорака у вештачкој интелигенцији, итд. Посебно важну класу детерминизационих алгоритама чине канонизациони алгоритми, који на излазу дају минимални комплетан детерминистички фази (тежински) аутомат еквивалентан улазном фази (тежинском) аутомату. Циљ дисертације је развој детерминизационих алгоритама базираних на концепту факторизације, као и препознавању и сажимању еквивалентних стања фази (тежинског) аутомата у конструкцији. При томе, препознавање и сажимање еквивалентних стања у случају фази аутомата врши се преко десно и лево инваријантних фази релација, а у случају тежинских аутомата преко десно и лево инваријантних Булових матрица. Примењена је техника уситњења партиција за добијање побољшаних алгоритама за рачунање највећих десно и лево инваријантних Булових матрица еквиваленција и матрица квази – уређења. На крају, разматрани су начини израчунавања највећих десно и лево инваријантних фази еквиваленција и фази квази – уређења када се алгоритми за њихово рачунање, базирани на техници уситњења партиција, не завршавају у коначном броју корака.
Научна област:	Рачунарске науке
Научна дисциплина:	Теорија израчунавања
Кључне речи:	Фази аутомати, фази језици, детерминизација
УДК:	519.71, 519.713, 519.76
CERIF класификација:	P110, P176
Тип лиценце Креативне заједнице:	**CC BY-NC-ND**

Data on Doctoral Dissertation

Doctoral Supervisor:	Miroslav D. Ćirić, Ph.D., full professor at Faculty of Sciences and Mathematics, University of Niš
Title:	Improved algorithms for determinization of fuzzy and weighted automata
Abstract:	Determinization algorithms are methods that calculate complete deterministic fuzzy (weighted) automaton that is language equivalent to the input fuzzy (weighted) automaton, and they have found application in numerous fields, including lexicographic analysis, analysis of regular expressions, automatic speech recognition, pattern recognition in artificial intelligence, etc. Especially important class of determinization algorithms are canonization algorithms, which produce minimal complete deterministic fuzzy (weighted) automaton equivalent to the input fuzzy (weighted) automaton. The aim of this dissertation is the development of determinization algorithms based on the concept of factorizations, as well as computing and merging of the indistinguishable states of fuzzy (weighted) automaton under construction. At the same time, computing and merging of the indistinguishable states is done by right and left invariant fuzzy relations in the case of fuzzy automata, as well as by right and left invariant Boolean matrices in the case of weighted automata. We apply the partition refinement technique to obtain improved algorithms for computing the greatest right and left invariant Boolean equivalence and quasi – order matrices. In the end, we consider ways to compute the greatest right and left invariant fuzzy equivalences and fuzzy quasi – orders when the algorithms for their computation, based on the partition refinement technique, are unable to stop in a finite number of steps.
Scientific Field:	Computer science
Scientific Discipline:	Theory of computing
Key Words:	Fuzzy automata, fuzzy languages, determinization
UDC:	519.71, 519.713, 519.76
CERIF Classification:	P110, P176
Creative Commons License Type:	**CC BY-NC-ND**

Zahvalnica

Najveću zahvalnost dugujem svom mentoru, prof. dr Miroslavu Ćiriću, na uključivanju na projekat "Razvoj metoda izračunavanja i procesiranja informacija: teorija i primene" (evidencioni broj 174013), kao i na kontinuiranoj pomoći, brojnim prijateljskim savetima i beskrajnoj podršci koju mi je pružio u toku doktorskih studija, od samih početaka prilikom odabira oblasti istraživanja, do samog formiranja svih naučnih rezultata prikazanih u tezi.

Prof. dr Jeleni Ignjatović takođe zahvaljujem na velikoj i prijateljskoj podršci koju je nesebično pružila kada je bilo najpotrebnije.

Veliku zahvalnost dugujem i svojim dragim prijateljima i kolegama, dr Ivani Micić, dr Zorani Jančić, kao i prof. dr Aleksandru Stamenkoviću, na nesebičnoj pomoći koju su mi pružili, bez koje disertacija ne bi imala oblik u kojem je sada.

Na kraju, ali ne i najmanje važno, zahvalnost dugujem svojoj porodici i prijateljima, posebno supruzi Aniti i roditeljima Svetlani i Predragu, koji su mi pružili neophodno razumevanje i podršku.

Sadržaj

Spisak algoritama

Skraćenice

DKA deterministički konačni automat. 1
FKA fazi konačni automat. 37
K-DFA krsip-deterministički fazi automat. 42
K-DFKA krsip-deterministički fazi konačni automat. 42
KDDFA kompletan deterministički dečji fazi automat. 76
KDFA kompletan deterministički fazi automat. 41, 70
KDFKA kompletan deterministički fazi konačni automat. 41
KDTA kompletan deterministički težinski automat. 43
KDTKA kompletan deterministički težinski konačni automat. 43
NKA nedeterministički konačni automat. 1
RSSRC reverzno slabo svojstvo reprezentativnih ciklusa. 98, 99
SRC svojstvo reprezentativnih ciklusa. 83
SSRC slabo svojstvo reprezentativnih ciklusa. 83, 85
TKA težinski konačni automat. 39
UOL uslov opadajućeg lanca. 10
URL uslov rastućeg lanca. 10

Predgovor

U klasičnoj teoriji automata, *determinizacija* podrazumeva proceduru nalaženja ekvivalentnog determinističkog konačnog automata (DKA, skraćeno) za dati nedeterministički konačni automat (NKA, skraćeno). Ona predstavlja osnovni problem u teoriji automata koji datira još iz rada Rabin i Scott [176] iz 1959. godine. Njihov originalni metod danas je poznat kao *podskupovna konstrukcija*. Poboljšanje ove metode, poznato pod nazivom *dostižna podskupovna konstrukcija*, danas je verovatno jedan od najistaknutijih primera determinizacionih metoda, a može se naći u većini klasičnih knjiga iz teorije automata [38, 92, 95, 130]. Dostižna podskupovna konstrukcija podrazumeva konstrukciju DKA-a u kojem je svako stanje skup stanja originalnog NKA-a koja su dostižna na trenutnoj poziciji. Za ulazni NKA sa n stanja, oba metoda mogu rezultirati DKA-om sa do 2^n stanja, zbog čega, sa teorijskog aspekta, ovi algoritmi spadaju u algoritme sa eksponencijalnom složenošću. Međutim, broj stanja rezultujućeg DKA retko dolazi u blizinu te gornje granice od 2^n, te je dostižna podskupovna konstrukcija poznata po dobroj efikasnosti u praksi [75, 74]. Sa druge strane, postoje i primeri kada je eksponencijalni rast broja stanja u oba algoritma neizbežan [75, 74], kao i primeri u kojima algoritmi troše puno memorije i rezultiraju DKA-om sa mnogo većim brojem stanja od minimalnog DKA-a, što je bila motivacija da van Glabbeek i Ploeger u [75, 74] razviju druge determinizacione metode za NKA-e koje rezultiraju ekvivalentnim DKA-ima sa još manjim brojem stanja od onih dobijeni primenom dostižne podskupovne konstrukcije.

Od svog uvođenja, determinizacija je našla primenu u brojnim oblastima, između ostalih u leksikografskoj analizi [201, Poglavlje 8], podudaranju uzoraka na osnovu regularnih izraza [69, Poglavlje 4], analizi proteinskih nizova [141], sintezi reaktivnih sistema [169], potvrdi probabilističkih sistema [50, 205], rasuđivanju o više agentnim sistemima [5], itd.

DKA-i su izražajni kao i NKA-i, u smislu da za svaki NKA postoji jezički ekvivalentan DKA. Za razliku od NKA-a, za svaki DKA postoji trivijalni, vremenski linearan, onlajn algoritam koji radi u konstantnom memorijskom prostoru preko kojeg se može proveriti da li se ulazni niz simbola može prihvatiti ili ne. Iz tog razloga, u mnogim praktičnim aplikacijama isplati se da se ulazni NKA konvertuje u ekvivalentan DKA, iako je, u najgorem slučaju, vremenska i prostorna složenost ove konverzije eksponencijalna u odnosu na broj stanja ulaznog NKA-a.

Posebno značajne determinizacione metode su one koje rezultiraju minimalnim DKA-om, i ovakve metode nazivamo *kanonizacionim metodama*. Verovatno najistaknutiji primer kanonizacionog metoda je *dvostruko reverzna procedura Brzozowskog*. Originalna ideja Brzozowkog u njegovom radu [29] bila je da se, za dati NKA $\mathcal{A}$, konstruiše minimalan DKA $d(r(d(r(\mathcal{A}))))$ ekvivalentan sa $\mathcal{A}$, pri čemu sa $r(\cdot)$ označavamo reverzni, a sa $d(\cdot)$ označavamo determinizacioni proces (na primer, prethodno spomenuta dostižna podskupovna konstrukcija). Uprkos eksponencijalnoj složenosti ovog algoritma, i njega odlikuje dobra efikasnost u većini konkretnih primera (videti [37, 200, 207]).

Mnogi istraživači proširili su osnovni model NKA-a. Takve generalizacije uključuju *težinske konačne automate* (TKA, skraćeno) i *fazi konačne automate* (FKA, skraćeno), u kojima prelazi, početna i završna stanja uzimaju vrednosti iz određenih struktura. U slučaju fazi automata, vrednosti za prelaze, početna i završna stanja nazivaju se istinitosne vrednosti, i uzimaju se iz određenih uređenih struktura. Koncept fazi automata potiče iz radova Santos [187] i Wee i Fu [210], a kasnije je adaptiran nad opštijim strukturama, a najćešće nad mrežno uređenim strukturama poput mrežno uređenih monoida [135], po-monoida [113], kompletnim distributivnim mrežama [14], generalnim mrežama [134], kao i kompletnim reziduiranim mrežama [97, 98, 171, 172, 174]. U slučaju težinskih automata, vrednosti za prelaze, početna i završna stanja nazivaju se težine, i uzimaju se iz opštijih struktura, na primer, poluprstena. Težinski automati nad poluprstenima izučavani su od strane mnogih autora u različitim kontekstima [60, 183]. Težinski automati nad posebnim tipovima poluprstena izučavani su u mnogim radovima, na primer, Mohri [154] nad tropskim poluprstenima, Droste i Kuich [59] nad hemiprstenima. Neki autori izučavali su i težinske automate i nad opštijim strukturama, na primer, nad jakim bimonoidima u [43, 61].

Za razliku od NKA-a, koji uvek mogu biti determinizovani, ne mogu svi FKA-i, niti TKA-i, biti determinizovani, te je problem determinizacije u ovakvim postavkama od posebnog značaja. Tokom godina, kroz mnoge radove izučavani su različiti metodi za determinizaciju FKA-a i TKA-a.

Determinizacija fazi automata prva je proučavana u radu Bělohlávek [14], za FKA-e nad kompletnim distributivnim mrežama, a u radu Li and Pedrycz [135] za FKA-e nad mrežno uređenim monoidima. Determinizacioni algoritmi prikazani u tim radovima generalizuju podskupovnu konstrukciju, a kao rezultat daju ekvivalentan *krisp-deterministički fazi automat* (k-DFA, skraćeno) za dati FKA. Ovi metodi imaju isti nedostatak kao i odgovarajući algoritam za NKA-e, a to je da neka stanja u rezultujućem k-DFA-u mogu biti nepotrebna. Stoga su Ignjatović i drugi u [97] primenili dostižnu podskupovnu konstrukciju za FKA-e nad kompletnim reziduiranim mrežama. Njihov algoritam rezultira k-DFA-om sa manjim brojem stanja od algoritama iz [14, 135]. Rezultujući k-DFA alternativno može biti konstruisan putem Nerodove desne kongruencije originalnog FKA-a, i koji se u [100] naziva Nerodov automat originalnog FKA-a. Kao što je napomenuto u [97], identična konstrukcija može biti urađena i u opštijem kontekstu, preciznije, za FKA-e nad mrežno uređenim monoidima, pa čak i za težinske automate nad poluprstenima.

Metod konstrukcije Nerodovog automata primenjen je i za TKA-e nad jakim bimonoidima u radu Ćirić i dr. [43]. Algoritam predložen u radu Jančić i dr. [109], koji generalizuje "prelaznu skupovnu konstrukciju" datu u [75, 74], rezultira krisp-determinističkim fazi ili težinskim automatom sa čak manjim brojem stanja od Nerodovog automata. Napominjemo da ovi radovi proširuju determinizacione metode za $(\max, \min)$-fazi automate date u [26, 27, 140, 157]. Sa druge tačke gledišta, determinizacione metode za težinske automate dali su Mohri [154] za težinske automate nad tropskim poluprstenom, kao i Kirsten i Mäurer [116] za težinske automate nad proizvoljnim poluprstenom. Njihov model determinističkog težinskog automata u ovoj disertaciji nazivamo *kompletan deterministički težinski automat* (KDTA).

S obzirom da su ovi metodi primenljivi na NKA-e, njihova složenost u najgorem slučaju, merena u pogledu veličine determinističkog fazi (težinskog) automata, ograničena je izrazom $2^{|A|\times m}$ za neki faktor m, pri čemu taj faktor zavisi od kardinalnosti skupa istinitosnih vrednosti (težina) dobijenog konstruisanjem determinističkog fazi (težinskog) automata. Xing i dr. su u [217] takođe ukazali na ovu činjenicu u analizi složenosti algoritama za L-fazi automate.

Jančić i dr. su u [110] dali algoritme za računanje k-DFA-a koji kombinuju determinizacioni metod sa metodom redukcije broja stanja. Ovi algoritmi, zapravo, istovremeno vrše determinizaciju i redukciju broja stanja, a efikasniji su od algoritama datih u [14, 97, 109, 135], u smislu da rezultiraju k-DFA-a sa manjim brojem stanja, a rade u istoj vremenskoj složenosti. Iako ovi algoritmi nužno ne rezultiraju minimalnim k-DFA-om, za razliku od kanonizacionih metoda kao determinizacioni metod tipa Brzozowski [108], ili determinizacioni metod pomoću jezičke inkluzije [147], ovi algoritmi mogu biti korišćeni unutar determinizacionog metoda tipa Brzozowski sa ciljem ubrzanja tog metoda.

Glavni nedostatak k-DFA-a je što nisu u stanju da raspoznaju fazi jezike sa beskonačnim rangom. Drugim rečima, svi determinizacioni metodi razvijeni u prethodno nabrojanim radovima ne mogu se primeniti na fazi automate koji raspoznaju fazi jezike beskonačnog ranga. Sa druge strane, takvi fazi automati nalaze primene u različitim oblastima. Na primer, $(\max, *)$-fazi automati, pri čemu $*$ označava neku t-normu, nalaze primenu u nekim problemima veštačke inteligencije u prepoznavanju uzoraka [8, 9], a takvi fazi automati raspoznaju fazi jezike sa beskonačnim rangom. Na primer, za datu reč $u \in X^*$, fazi jezik $L_u : X^* \to [0, 1]$ definisan sa $L_u(v) = 10 - d(u, v)$, za svako $v \in X^*$, pri čemu $d(u, u)$ označava Levenshteinovo rastojanje između reči u i v, može se koristiti za modelovanje pojma "reči koje su slične sa u", koji je neodređen, odnosno nejasan po svojoj prirodi. Fazi jezik L_u ima beskonačan rang i ne može biti raspoznat nijednim k-DFA, međutim, postoji $(\max, *)$-fazi automat koji ga raspoznaje [9]. Fazi automati i fazi jezici sa beskonačnim rangom imaju širok spektar primene, uključujući oblasti poput leksičke analize, opisa prirodnih i programskih jezika, sistema zasnovanim na znanju, kontrolnih sistema, neuronskih mreža, prepoznavanje uzoraka, klinički nadzor, baze podataka, diskretni sistemi događaja, i tako dalje (videti [22, 28, 63, 70, 83, 157, 162, 165, 175, 179, 209, 218] i reference u njima).

Stoga, de Mendívil i Garitagoitia uvode u [145] takozvani *kompletan deterministički fazi automat* (KDFA, skraćeno), koji predstavlja generalizaciju k-DFA-a, i koji je u stanju da raspozna fazi jezik beskonačnog ranga. U istom radu, autori su dali takozvanu *Lemu o napumpavanju* za fazi jezike raspoznate od KDFA-a, te time odredili neophodan uslov za determinizaciju fazi automata. U radu [144], isti autori dali su determinizacioni algoritam za konverziju fazi automata u ekvivalentan KDFA.

Glavni zadatak ove disertacije je razvoj algoritama za determinizaciju fazi i težinskih automata, pri čemu pod determinizacijom fazi (težinskih) automata podrazumevamo konverziju FKA (TKA) u jezički ekvivalentan KDFA (KDTA). Svi predloženi algoritmi zasnivaju se na dva koncepta. Prvi koncept predstavlja *faktorizaciju fazi skupova* u slučaju fazi automata, odnosno *faktorizaciju vektora* u slučaju težinskih automata. Faktorizacija je dobro poznati koncept, koji je najpre uveden u radu Kirsten i Mäurer [116] u kontekstu težinskih automata, a kasnije adaptiran u radu de Mendívil i Garitagoitia [144] u kontekstu fazi automata. Faktorizacija ima dvostruku ulogu: ona se koristi kako u računanju stanja kompletnog deterministčkog fazi (težinskog) automata u konstrukciji, tako i u računanju težina prelaza između stanja. Drugi koncept predstavlja izračunavanje i sažimanje ekvivalentnih stanja fazi (težinskog) automata u konstrukciji. Na ovaj način, poboljšavaju se algoritmi za determinizaciju fazi i težinskih automata razvijeni u [109, 110, 116, 143, 144, 145, 154] (videti i reference u ovim radovima).

Izračunavanje i sažimanje ekvivalentnih stanja u slučaju fazi automata vrši se uz pomoć *desno i levo invarijantnih fazi relacija*, koje su vrlo dobro izučavane u skorijoj literaturi, primenjivane u determinizaciji i redukciji stanja FKA, i u uskoj su vezi sa simulacijama i bisimulacijama na FKA (videti [44, 45, 41, 42, 110, 147, 197, 198, 195,

193, 211]). Sa druge strane, koncept desno i levo invarijantnih fazi relacija nije direktno primenljiv na težinske automate, niti je postojao sličan pokušaj adaptacije tih pojmova iz dva glavna razloga. Prvi razlog je što strukture za težine koje uzimaju težinski automati nisu uređene, što je neophodno za postojanje najveće takve desno ili levo invarijantne relacije. Drugi razlog je nepostojanje neke operacije "ostatka" ili "reziduala" koja postoji u istinitosnim strukturama u slučaju fazi automata, koja je neophodna za računanje najveće takve desno ili levo invarijantne relacije. Kao što ćemo pokazati, ovi nedostaci mogu se na određeni način nadomestiti u aditivno idempotentnim poluprstenima. Preciznije, u ovakvim strukturama definišemo takozvane *desno i levo invarijantne Boolove matrice*, gde smo u mogućnosti da obezbedimo postojanje i računanje najvećih takvih matrica.

Disertacija je organizovana na sledeći način. U Glavi 1 uvedeni su osnovni pojmovi koji će biti neophodni za izlaganje glavnih rezultata. Najveća pažnja posvećena je mrežno uređenim strukturama, a posebno (kompletnim) reziduiranim mrežama koje služe kao struktura istinitosnih vrednosti za fazi automate, kao i dioidima (delimično uređenim poluprstenima) i aditivno idempotentnim poluprstenima koji služe kao struktura težina za težinske automate. Takođe, posebna pažnja posvećena je i reziduiranim mrežama na realnom jediničnom intervalu $[0, 1]$ i takozvanim BL-algebrama na tom intervalu.

U Glavi 2 dajemo definicije fazi i težinskih automata, kao i fazi jezika i formalnih stepenih redova. S obzirom da uslovi za završetak prezentovanih determinizacionih algoritama direktno zavise od interne strukture fazi (težinskih) automata, dajemo neophodne oznake i pojmove vezane za grafičku reprezentaciju fazi (težinskih) automata, poput putanja i ciklusa. Na kraju, dajemo definiciju kompetno determinističkih fazi (težinskih) automata, kao i krisp-determinističkih fazi automata kao poseban slučaj prethodnih.

Glava 3 posvećena je računanju najvećih desno i levo invarijantnih fazi ekvivalencija i fazi kvazi-uređenja. Motivisani činjenicom da su brži algoritmi za računanje najveće bisimulacione ekvivalencije razvijeni na osnovu njene uske veze sa problemom relacione najgrublje particije (videti [57, 73, 114, 178]), u Glavi 3 dati su algoritmi za računanje najvećih desno i levo invarijantnih fazi ekvivalencija i fazi kvazi-uređenja na osnovu *tehnike usitnjenja particija*. Tehnika usitnjenja particija dobro je poznata ne samo u teoriji automata (videti Hopcroft [91]), već i u brojnim oblastima računarskih nauka, poput teorije grafova, stringova i Boolovih matrica (videti [84]). Nažalost, algoritmi za računanje najvećih desno i levo invarijantnih fazi ekvivalencija i fazi kvazi-uređenja bazirani na tehnici usitnjenja particija ne pružaju ubrzanje u odnosu na iste algoritme bazirani na Knaster–Tarski teoremi o fiksnoj tački koji su prethodno razvijeni u [44, 45, 197], odnosno, rade u istoj vremenskoj složenosti. Sa druge strane, pokazujemo da su nizovi fazi ekvivalencija (odnosno fazi kvazi-uređenja), generisani preko algoritama baziranih na tehnici usitnjenja particija, konvergentni u slučaju da je struktura istinitosnih vrednosti BL-algebra na realnom jediničnom intervalu $[0, 1]$, te da se najveća desno (levo) invarijantna fazi ekvivalencija (fazi kvazi-uređenje) može dobiti preko granične vrednosti generisanih nizova. Razlog leži u tome što u BL-algebrama na intervalu $[0, 1]$ postoji veza između neprekidnosti t-norme $\otimes$ i neprekidnosti $\otimes$ kao funkcije realne promenljive, što nam omogućuje da koristimo graničnu vrednost u uobičajenoj topologiji na $[0, 1]$. Na kraju, pokazano je da se brži algoritmi mogu dobiti u slučaju kada se računaju najveće desno ili levo invarijantne *krisp* ekvivalencije na fazi automatu. Naime, dok algoritmi za računanje najveće desno ili levo invarijantne krisp ekvivalencije na fazi automatu razvijeni u [44, 45] rade u $\mathcal{O}(mn^5)$ vremenu, pri čemu je m broj simbola ulaznog alfabeta, a n broj stanja ulaznog fazi automata, algoritmi razvijeni u ovoj

glavi rade u $\mathcal{O}(mn^3)$ vremenu.

U Glavi 4 dati su algoritmi za determinizaciju fazi automata. Najpre je uveden koncept faktorizacije fazi podskupova u kompletnim reziduiranim mrežama, a zatim su ispitivana osnovna svojstva faktorizacije. Potom je razvijen determinizacioni metod baziran na korišćenju faktorizacije fazi podskupova i desno invarijantnih fazi kvazi-uređenja. Ovaj metod svodi se na konstrukciju fazi automata dobijenog sažimanjem stanja primenom desno invarijantih fazi kvazi-uređenja, a potom primenom metoda koji su razvili de Mendívil i Garitagoitia [145]. Na ovaj način omogućujemo determinizaciju u slučajevima kada direktna primena metoda iz [145] rezultira KDFA-om sa beskonačnim brojem stanja. Takođe, razvijen je i metod za konstrukciju dečjeg fazi automata, koji zapravo kombinuje determinizaciju preko skupova prelaza i sažimanje stanja fazi automata, čime se dobijaju dodatna poboljšanja. Na kraju, definišemo takozvano *slabo svojstvo reprezentativnih ciklusa* i dokazujemo da je to svojstvo potreban i dovoljan uslov da se prikazani algoritmi završe u konačnom broju koraka. Ovo svojstvo opštije je od *svojstva reprezentativnih ciklusa* prethodno određenog u [144] kao potreban i dovoljan uslov za determinizaciju preko maksimalne faktorizacije, čime proširujemo klasu fazi automata koji mogu biti determinizovani. Ovo svojstvo formuliše se jedino na osnovu interne strukture ulaznog fazi automata.

Glava 5 posvećena je razvoju algoritma za kanonizaciju fazi automata. Dati algoritam predstavlja adaptaciju dobro poznatog dvostruko reverznog metoda Brzozowskog, i bazira se na korišćenju faktorizacije fazi podskupova i levo invarijantnih fazi kvazi-uređenja. Algoritam rezultira minimalnim KDFA-om, čime se poboljšavaju algoritmi razvijeni u Glavi 4. *Reverzno slabo svojstvo reprezentativnih ciklusa* određen je kao potreban i dovoljan uslov da bi se dati kanonizacioni metod završio u konačnom broju koraka, pod uslovom da je u algoritmu korišćena maksimalna faktorizacija. Nažalost, primerom je pokazano da postoje fazi automati koji ne zadovoljavaju reverzno slabo svojstvo reprezentativnih ciklusa, ali zadovoljavaju slabo svojstvo reprezentativnih ciklusa, što znači da postoje situacije kada je moguće koristiti algoritme iz Glave 4, a nije moguće primeniti algoritam opisan u Glavi 5.

U Glavi 6 dati su algoritmi za determinizaciju težinskih automata. Najpre su date definicije (slabo) desno i levo invarijantnih Boolovih matrica kao rešenja određenih sistema matričnih jednačina i nejednačina nad aditivno idempotentnim poluprstenima. Potom su dati algoritmi za računanje najvećih desno i levo invarijantnih Boolovih matrica ekvivalencije, kao i matrica kvazi-uređenja. Algoritmi su takođe bazirani na tehnici usitnjenja particije. Stoga, algoritmi za računanje najvećih desno i levo invarijantnih Boolovih matrica ekvivalencije izvršavaju se brže od algoritama za računanje najveće simulacije razvijenih u [51], a koji se mogu direktno adaptirati za računanje najvećih desno i levo invarijantnih Boolovih matrica ekvivalencije. Potom je pokazano kako se algoritmi razvijeni u Glavama 3 i 4 mogu primeniti na težinske automate nad komutativnim, aditivno idempotetnim poluprstenima bez delioca nule. Na taj način, poboljšavaju se algoritmi razvijeni u [109, 116, 154] za determinizaciju težinskih automata.

Implementacija prikazanih algoritama vršena je u programskom jeziku C#. Odgovarajući kodovi prikazani su u Dodatku A.

Glava 1

Uvodni pojmovi i rezultati

U ovoj glavi prikazani su osnovni pojmovi i rezultati koji se koriste u daljem izlaganju. Na dobro poznate oznake i rezultate se samo pozivamo.

1.1 Skupovi, relacije i uređeni skupovi

U ovoj sekciji dat je opsežan pregled osnovnih algebarskih pojmova, kao što su skupovi i relacije, s obzirom da se pojmovi fazi skupova i fazi relacija direktno oslanjaju na ove pojmove. Dodatni pojmovi i terminologija od značaja za dalje izlaganje, kao što su funkcije, particije i uređenja, takođe su opsežno opisani. Pojmovi i terminologija iz ove sekcije u skladu su sa Bělohlávek [15], Bělohlávek i Vychodil [16], Birkhoff i Barti [21], Clark [47], Grätzer [81].

Pojam *skup* koristi se kao u intuitivnoj teoriji skupova, odnosno, kao kolekcija objekata koje nazivamo elementima ili članovima skupa. Skup je implicitno snabdeven identičkom relacijom $=$ koja obezbeđuje trivijalnu informaciju o tome koji elementi su jednaki a koji nisu, i ova informacija dolazi apriorno pojmu skupa. Sa $=, \subseteq, \cap, \cup$ i $-$ označava se jednakost, inkluzija, presek, unija i razlika između dva skupa, respektivno. Skup prirodnih brojeva bez nule označava se sa $\mathbb{N}$, dok se skup realnih brojeva označava sa $\mathbb{R}$. Takođe se koristi i oznaka $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Neka su $A_1, A_2, \ldots, A_n$, $n \in \mathbb{N}$, neprazni skupovi. *Relacija* između skupova $A_1, A_2, \ldots, A_n$ je bilo koji podskup R Dekartovog proizvoda $A_1 \times A_2 \times \ldots \times A_n$. U slučaju $n = 2$, R se naziva *binarna relacija*, a kada je pri tome i $A_1 = A_2 = A$, tada se binarna relacija između A_1 i A_2 naziva binarna relacija na A. Za binarnu relaciju R na A, pišemo xRy umesto $(x, y) \in R$. Neke binarne relacije na nepraznom skupu A od posebnog značaja su *prazna relacija* u uobičajenoj oznaci $\emptyset$, *relacija jednakosti* (ili *dijagonalna relacija* ili *identička relacija*) $\triangle_A$ definisana sa $\triangle_A = \{(x, x) | x \in A\}$, kao i *univerzalna relacija* (ili *puna relacija*) ∇_A definisana sa $\nabla_A = A \times A$. U slučaju da je skup A jasan iz konteksta, izostavlja se A iz gornjih oznaka i piše se jednostavno $\triangle$ i ∇ umesto $\triangle_A$ i ∇_A, respektivno. S obzirom da se binarna relacija na skupu A definiše kao podskup Dekartovog proizvoda $A \times A$, jednakost, inkluzija, unija, presek i razlika binarnih relacija na skupu A definišu se kao za poskupove skupa $A \times A$. Za binarnu relaciju R na skupu A, definiše se njen *inverz* kao binarna relacija R^{-1} na A sa $R^{-1} = \{(x, y) \in A \times A | (y, x) \in R\}$, dok se njen *rang* definiše kao podskup $\mathrm{rang}(R)$ od A sa $\mathrm{rang}(R) = \{y \in A | (x, y) \in R, \text{ za neko } x \in A\}$.

Neka je A neprazan skup, $B, C \subseteq A$ podskupovi od A, i neka su $R, S \subseteq A \times A$ binarne relacije na A. Tada se definiše *proizvod* (ili *kompozicija*) $R \circ S$ od R i S kao binarna relacija na A definisana sa

$$R \circ S = \{(x, z) \in A \times A | (\exists y \in A)(x, y) \in R \text{ i } (y, z) \in S\}, \tag{1.1}$$

kao i *proizvodi* $B \circ R$ i $R \circ B$ kao podskupovi od A definisani sa

$$B \circ R = \{y \in A | (\exists x \in A) x \in B \text{ i } (x, y) \in R\}, \tag{1.2}$$
$$R \circ B = \{x \in A | (\exists y \in A)(x, y) \in R \text{ i } y \in A\}, \tag{1.3}$$

respektivno. Na kraju, *proizvod* poskupova B i C definisan je sa

$$B \circ C = \begin{cases} 1, & B \cap C \neq \emptyset, \\ 0, & \text{inače.} \end{cases} \tag{1.4}$$

Drugim rečima, $B \circ C$ je istinitosna vrednost tvrđenja $B \cap C \neq \emptyset$.

Propozicija 1.1. *Neka je A neprazan konačan skup, i neka je $B, C \subseteq A$, $R, S, T \subseteq A \times A$ i $\{S_i\}_{i \in I} \subseteq A \times A$. Tada važe sledeća svojstva:*

$$(R \circ S) \circ T = R \circ (S \circ T), \tag{1.5}$$
$$B \circ (R \circ S) = (B \circ R) \circ S, B \circ (R \circ C) = (B \circ R) \circ C, (R \circ S) \circ B = R \circ (S \circ B), \tag{1.6}$$
$$S_1 \subseteq S_2 \quad \textit{povlači} \quad R \circ S_1 \subseteq R \circ S_2 \quad i \quad S_1 \circ R \subseteq S_2 \circ R, \tag{1.7}$$
$$R \circ \left(\bigcup_{i \in I} S_i \right) = \bigcup_{i \in I} (R \circ S_i), \quad \left(\bigcup_{i \in I} S_i \right) \circ R = \bigcup_{i \in I} (S_i \circ R), \tag{1.8}$$
$$(R \circ S)^{-1} = S^{-1} \circ R^{-1}, \tag{1.9}$$

Stoga, sve zagrade u (1.5) i (1.6) mogu da se izostave.

Neka su A i B neprazni skupovi. *Funkcija* (ili *preslikavanje*) iz skupa A u skup B je binarna relacija $f \subseteq A \times B$ takva da važi:

(1) za svako $x \in A$ postoji $y \in B$ tako da je $(x, y) \in f$;

(2) ako je $(x, y_1) \in f$ i $(x, y_2) \in f$, tada je $y_1 = y_2$.

Ako je f preslikavanje iz skupa A u skup B, koristi se oznaka $f : A \to B$ umesto $f \subseteq A \times B$, dok se koristi oznaka $f(x) = y$ umesto $(x, y) \in f$. Preslikavanje f naziva se *injektivno* (ili *injekcija*, ili *jedan-na-jedan*) ako $x_1 \neq x_2$ povlači $f(x_1) \neq f(x_2)$, *surjektivno* (ili *surjekcija*, ili *na*) ako za svako $y \in B$ postoji $x \in A$ tako da je $y = f(x)$, *bijektivno* (ili *bijekcija*) ako je istovremeno i injektivno i surjektivno. Za preslikavanja $f : A \to B$ i $g : B \to C$, njihov *proizvod* (ili *kompozicija*) je preslikavanje $f \circ g : A \to C$ definisano sa $(f \circ g)(x) = f(g(x))$.

Skup svih preslikavanja iz skupa A u skup B označava se sa B^A. U slučaju kada je $B = \{0, 1\}$, skup B obično se obeležava sa 2 (odnosno, stavlja se $2 = \{0, 1\}$, bez mogućnosti mešanja sa celim brojem 2), i svaki $\chi \in 2^X$, pri čemu je $X \subseteq A$, naziva se *karakteristična funkcija* od X. Postoji prirodna bijekcija između skupa 2^A i skupa svih podskupova skupa A. Zaista, za svaki $X \subseteq A$ definiše se $\chi_X : A \to \{0, 1\}$ sa $\chi_X(x) = 1$ ako je $x \in X$, kao i $\chi_X(x) = 0$ ako $x \notin X$. Sa druge strane, za svaku funkciju $\chi \in 2^A$ definiše se podskup X_χ od A sa $X_\chi = \{x \in A | \chi(x) = 1\}$. Pored toga, važi $X_{\chi_X} = X$ i $\chi_{X_\chi} = \chi$, za svaki $X \subseteq A$. Iz tog razloga, ne pravi se razlika između podskupova od A i njihovih odgovarajućih karakterističnih funkcija, te se i skup svih podskupova od A takođe označava sa 2^A.

Za binarnu relaciju R na A kaže se da je:

(1) *refleksivna*, ako $(x, x) \in R$, za svako $x \in A$;

(2) *simetrična*, ako $(x, y) \in R$ povlači $(y, x) \in R$, za svako $x, y \in A$;

(3) *anti-simetrična*, ako $(x, y) \in R$ i $(y, x) \in R$ povlači $x = y$, za svako $x, y \in A$;

(4) *tranzitivna*, ako $(x,y) \in R$ i $(y,z) \in R$ povlači $(x,z) \in R$, za svako $x,y,z \in A$.

Binarna relacija R na A naziva se *kvazi-uređenje* ako je refleksivna i tranzitivna. U skorijoj literaturi, takođe se koristi i naziv "*preuređenje*" umesto "kvazi-uređenje" (videti klasične udžbenike [80, 181, 190] i radove [35, 36, 103, 104]), ali mi se opredeljujemo za originalan naziv "kvazi-uređenje" koji se češće koristi u literaturi vezanoj za mreže (videti Birkhoff [20], Davey i Priestley [52]).

Binarna relacija E na A naziva se *ekvivalencija* ako je simetrično kvazi-uređenje. Za ekvivalenciju E na A i element $x \in A$, skup $E_x = \{y \in A | (x,y) \in E\}$ naziva se *klasa ekvivalencije* elementa x u odnosu na E. Skup svih klasa ekvivalencija $A/E = \{E_x | x \in A\}$ naziva se *faktor skup* (ili *količnik skup*) skupa A u odnosu na E.

Particija skupa A je bilo koji skup $\pi = \{B_1, B_2, \ldots, B_m\}$, $m \in \mathbb{N}$, takav da važi:

(1) $B_i \subseteq A$ i $B_i \neq \emptyset$ za svako $1 \leq i \leq m$;

(2) $B_1 \cup B_2 \cup \ldots \cup B_m = A$.

(3) $B_i \cap B_j = \emptyset$, za svako $1 \leq i < j \leq m$;

Svaki $B_k \in \pi$, $1 \leq k \leq m$, naziva se *blok* particije π. Uslov (1) znači da je svaki blok particije neprazan podskup skupa A, uslov (2) znači da je unija svih blokova jednaka skupu A, dok uslov (3) znači da se različiti blokovi ne seku. Za svaki neprazan skup A postoji bijektivna korespodencija između ekvivalencija na skupu A i particija skupa A. Zaista, za svaku ekvivalenciju E na A, skup $\pi_E = A/E$ je particija skupa A. Takođe, za svaku particiju $\pi = \{B_1, B_2, \ldots, B_m\}$ skupa A, binarna relacija $E_\pi = \{(x,y) | x,y \in B_k,$ za neko $1 \leq k \leq m\}$, je ekvivalencija na A. Štaviše, važi $E = E_{\pi_E}$ i $\pi = \pi_{E_\pi}$. Stoga, ne pravimo razliku između particija skupa i njihovih odgovarajućih ekvivalencija.

Neka su π i μ dve particije skupa A. Kaže se da je particija μ *usitnjenje* particije π ako za svaki blok $B_\mu \in \mu$ postoji blok $B_\pi \in \pi$ takav da je $B_\mu \subseteq B_\pi$. Ako je μ usitnjenje π, tada je $E_\mu \subseteq E_\pi$, tj. ekvivalencija koja odgovara particiji μ sadržana je u ekvivalenciji koja odgovara particiji π.

Binarna relacija na skupu A naziva se *delimično uređenje* ako je anti-simetrično kvazi-uređenje. Ako je $\leq$ delimično uređenje, tada se koristi oznaka $x \leq y$ umesto $(x,y) \in \leq$, kao i $x < y$ u slučaju kada je $x \leq y$ i $x \neq y$, za $x,y \in A$. Uz to, $(A, \leq)$ naziva se *delimično uređeni skup*. Za svaka dva elementa x,y delimično uređenog skupa $(A, \leq)$, ako je $x \leq y$ ili $y \leq x$, kaže se da su x i y *uporedivi*, a inače se kaže da su *neuporedivi*. Reč "delimično" u nazivima "delimično uređenje" ili "delimično uređen skup" ukazuje da ne moraju svi parovi elemenata skupa A biti uporedivi. U slučaju da su svi parovi elemenata skupa A uporedivi, tada se $\leq$ naziva *linearno uređenje*, dok se par $(A, \leq)$ naziva *linearno uređenjeni skup* ili *lanac*.

Za preslikavanje $f : A \to B$, pri čemu su A i B uređeni skupovi, kaže se da je *izotono* ili *rastuće* ako $x \leq y$ povlači $f(x) \leq f(y)$, za svako $x,y \in A$. Slično, za preslikavanje f kaže se da je *antitono* ili *opadajuće* ako $x \leq y$ povlači $f(y) \leq f(x)$, za svako $x,y \in A$. Za f se kaže da je *izomorfizam* uređenih skupova A i B, ili da je *izomorfno*, ako je f bijektivno preslikavanje, a f i f^{-1} izotona preslikavanja.

Neka je $(A, \leq)$ delimično uređen skup. Za element $x \in A$ kaže se da je *minimalan* element skupa A ako za svaki $y \in A$ takav da je $y \leq x$ važi $x = y$, odnosno, ako u A ne postoji element strogo manji od njega. Slično, x je *maksimalan* element skupa A ako za svaki $y \in A$ takav da je $x \leq y$ važi $x = y$, odnosno, ako u A ne postoji element strogo veći od njega. Osim toga, x je *najmanji* element skupa A ako za svako $y \in A$ važi $x \leq y$, odnosno, ako je manji od svakog drugog elementa iz A, dok je x *najveći* element skupa A ako za svako $y \in A$ važi $y \leq x$, odnosno, ako je veći od svakog drugog elementa iz A. Najmanji (odnosno, najveći) element skupa, ukoliko

postoji, je jedinstven, i obično se najmanji element skupa označava sa 0, dok se najveći element skupa označava sa 1 (ukoliko svaki od njih postoji). U slučaju da skup ima najmanji (odnosno, najveći) element, tada je on istovremeno i jedinstveni minimalni (odnosno, maksimalni) element skupa. Inače, u skupu može postojati više od jednog minimalnog (odnosno, maksimalnog) elementa.

Neka je $(A, \leq)$ delimično uređen skup i neka je $B \subseteq A$. Za element $x \in A$ kaže se da je *donja granica* skupa B, ako je $x \leq y$, za svako $y \in B$. Slično, element $x \in A$ naziva se *gornja granica* skupa B, ako je $y \leq x$, za svako $y \in B$. Skup svih donjih granica skupa B označava se sa $\mathcal{L}(B)$, dok se skup svih gornjih granica skupa B označava sa $\mathcal{U}(B)$. Ako postoji najveći element skupa $\mathcal{L}(B)$, tada se taj element naziva *najveća donja granica* ili *infimum*. Dualno, ako postoji najmanji element skupa $\mathcal{U}(B)$, tada se taj element naziva *najmanja gornja granica* ili *supremum*[1]. Infimum skupa B, ukoliko postoji, označava se sa $\bigwedge B$ (ili $\inf B$), dok se supremum skupa B, ukoliko postoji, označava sa $\bigvee B$ (ili $\sup B$). U slučaju da je $B = \{x_i | i \in I\}$, tada umesto oznaka $\bigwedge B$ i $\bigvee B$ koristimo oznake, redom,

$$\bigwedge_{i \in I} x_i \quad \text{i} \quad \bigvee_{i \in I} x_i,$$

a ukoliko je I konačan skup od n elemenata i važi $I = \{1, 2, \ldots, n\}$, tada se umesto gornjih oznaka koristi, redom,

$$\bigwedge_{i=1}^{n} x_i \quad \text{i} \quad \bigvee_{i=1}^{n} x_i.$$

Napomenimo da, ukoliko 0 postoji u A, tada važi $\bigwedge A = 0 = \bigvee \emptyset$, i dualno, ukoliko 1 postoji u A, tada je $\bigvee A = 1 = \bigwedge \emptyset$.

Delimično uređen skup $(A, \leq)$ naziva se *infumum-polumreža* ako za svaki dvoelementni podskup (stoga, i za svaki neprazan konačan podskup) skupa A postoji infimum, dok se naziva *supremum-polumreža* ako za svaki dvoelementni podskup skupa A postoji supremum. Štaviše, A se naziva *kompletna infumum-polumreža* (odnosno, *kompletna supremum-polumreža*) ako za svaki podskup skupa A postoji infimum (odnosno, supremum). Pored toga, za A se kaže da je *(kompletna) mreža* ako je istovremeno (kompletna) infumum-polumreža i (kompletna) supremum-polumreža. Ako u mreži A postoji i najmanji element 0, kao i najveći element 1, tada se mreža A naziva *ograničena mreža*, i označava sa $(A, \leq, 0, 1)$.

Neka je $(A, \leq)$ delimično uređen skup. Kaže se da A zadovoljava *uslov rastućeg lanca (URL)*, ako je svaki strogo rastući niz elemenata skupa A konačan, ili ekvivalentno, za bilo koji niz

$$x_1 \leq x_2 \leq x_3 \leq \ldots$$

postoji $n \in \mathbb{N}$ tako da je $x_{n+k} = x_n$ za svako $k \in \mathbb{N}_0$. Slično, kaže se da A zadovoljava *uslov opadajućeg lanca (UOL)*, ako je svaki strogo opadajuću niz elemenata skupa A konačan, ili ekvivalentno, za bilo koji niz

$$\ldots \leq x_3 \leq x_2 \leq x_1$$

postoji $n \in \mathbb{N}$ tako da važi $x_{n+k} = x_n$ za svako $k \in \mathbb{N}_0$.

[1] U engleskoj literaturi, česti nazivi za infimum i supremum su *meet* i *join*, respektivno.

1.2 Monoidi, poluprsteni i mreže

U ovoj sekciji dat je pregled osnovnih algebarskih struktura koje su neophodne za razumevanje struktura za fazi i težinske automate. Najpre se daju definicije i primeri monoida i poluprstena, da bi kasnije bili prikazani algebarska definicija mreža, osobine mreža, kao i odgovarajući primeri mreža. Oznake i terminologija u skladu su sa sledećim knjigama: Birkhoff [20], Blyth [23], Davey i Priestley [52], Golan [76], Grätzer [80] i Roman [181].

Neka je S neprazan skup i $n \in \mathbb{N}$ prirodan broj. Svako preslikavanje oblika $f : S^n \to S$ naziva se *operacija* na skupu S. U slučaju $n = 2$, operacija f naziva se *binarna operacija* na S, dok se u slučaju $n = 1$ naziva *unarna operacija* na S. Za binarnu operaciju $\cdot$ na S obično se koristi oznaka $x \cdot y = z$ umesto $\cdot(x, y) = z$. Za binarnu operaciju $\cdot$ na S kaže se da je:

(1) *asocijativna*, ako zadovoljava *zakon asocijativnosti*:

$$(x \cdot y) \cdot z = x \cdot (y \cdot z), \quad \text{za svako } x, y, z \in S;$$

(2) *komutativna*, ako zadovoljava *zakon komutativnosti*:

$$x \cdot y = y \cdot x, \quad \text{za svako } x, y \in S;$$

(3) *idempotentna*, ako zadovoljava *zakon idempotencije*:

$$x \cdot x = x, \quad \text{za svako } x \in S.$$

Polugrupa je uređeni par $(S, \cdot)$, pri čemu je S neprazan skup, a $\cdot$ asocijativna binarna operacija, koja se naziva *proizvod*. Polugrupa je *komutativna* ukoliko je proizvod komutativna binarna operacija. Slično, polugrupa je *idempotentna* ukoliko je proizvod idempotentna binarna operacija.

Monoid je uređena trojka $(S, \cdot, 1)$ koja se sastoji od nepraznog skupa S, binarne operacije $\cdot$ na S, kao i elementa $1 \in S$ tako da važi:

(1) $(S, \cdot)$ je polugrupa;

(2) 1 je *jedinični element*, odnosno važi

$$1 \cdot x = x \cdot 1 = x, \quad \text{za svako } x \in S.$$

Primetimo da je jedinični element monoida jedinstven.

Najistaknutiji primer monoida koji se koristi u ovom izlaganju je *monoid reči*. Neka je X neprazan konačan skup koji nazivamo *alfabet*, dok se svaki njegov element $x \in X$ naziva *slovo* ili *simbol*. Tada je *reč nad alfabetom* X (ili jednostavno *reč*) bilo koja konačna n-torka $\omega = (x_1, x_2, \ldots, x_n)$, pri čemu je $x_i \in X$ za svako $i \in \{1, 2, \ldots, n\}$ i $n \in \mathbb{N}_0$. Reči označavamo jednostavno sa $\omega = x_1 x_2 \ldots x_n$ umesto $\omega = (x_1, x_2, \ldots, x_n)$. Ceo broj n naziva se *dužina* reči ω, a označava se sa $|\omega|$. Sa ε označava se *prazna reč*, odnosno jedinstvena reč dužine 0. Skup svih reči nad alfabetom X označava se sa X^*. Na skupu X^* definiše se binarna operacija *konkatenacije* ili *spajanja* sa $\omega_1 \cdot \omega_2 = x_1 x_2 \ldots x_n y_1 y_2 \ldots y_m$, gde je $\omega_1 = x_1 x_2 \ldots x_n \in X^*$ i $\omega_2 = y_1 y_2 \ldots y_m \in X^*$. Tada struktura $(X^*, \cdot, \varepsilon)$ predstavlja monoid, koji se naziva *slobodan monoid nad* X ili *monoid reči*. Sa $X^+ = X^* - \{\varepsilon\}$ označava se *slobodna polugrupa nad* X ili *polugrupa reči*. *Jezik nad alfabetom* X je svaki podskup od X^*. Za dve reči $u, v \in X^*$ kaže se da je *u prefiks* od v ako postoji reč $w \in X^*$ takva da je $v = uw$.

Za monoid $(S, \cdot, 1)$ kaže se da je *komutativan* ako je $(S, \cdot)$ komutativna polugrupa, i da je *idempotentan* ako je $(S, \cdot)$ idempotentna polugrupa. Komutativni monoid često

se označava sa $(S, +, 0)$, odnosno, proizvod se obično naziva *sabiranje* i označava sa $+$, a jedinični element se označava sa 0 i naziva *nula*.

Neka je $(S, \cdot, 1)$ monoid. *Podmonoid* je bilo koji podskup $H \subseteq S$ takav da za sve $x, y \in H$ važi $x \cdot y \in H$, kao i $1 \in H$. Primetimo da je svaki podmonoid nekog monoida i sam monoid. Za dati podskup $H \subseteq S$ monoida S, *monoid generisan sa* H, u oznaci $\langle\langle H \rangle\rangle$, je najmanji podmonoid od S koji sadrži H (u odnosu na inkluziju skupova), i može se dobiti presekom svih podmonoida od S koji sadrže H. Pored toga, ako se za svako $x \in S$ i $n \in \mathbb{N}_0$ definiše x^n rekurzivno sa

$$x^n = \begin{cases} 1, & n = 0 \\ x^{n-1} \cdot x, & \text{inače} \end{cases},$$

tada se $\langle\langle H \rangle\rangle$ može predstaviti kao

$$\langle\langle H \rangle\rangle = \{x_1^{n_1} \cdot x_2^{n_2} \cdot \ldots \cdot x_m^{n_m} \mid m \in \mathbb{N} \text{ i } x_k \in H,\ n_k \in \mathbb{N}_0, \text{ za svako } 1 \leq k \leq m\}.$$

Kažemo da je S *generisan* sa H (ili da je H *sistem generatora* za S) kada je $S = \langle\langle H \rangle\rangle$, kao i da je S *minimalno generisan* (ili da je H *minimalan sistem generatora* za S) kada je S generisan sa H i ne postoji nijedan pravi podskup skupa H takav da je S generisan sa tim podskupom. Na kraju, kažemo da je S *konačno generisan* ako S ima konačan sistem generatora. Za monoid $(S, \cdot, 1)$ kaže se da je *lokalno konačan* ako je svaki njegov konačno generisan podmonoid konačan (za više detalja, videti [182]). Jasno je da je svaki komutativan i idempotentan monoid lokalno konačan.

Poluprsten je uređena petorka $(S, +, \cdot, 0, 1)$ koja se sastoji od nepraznog skupa S, dve binarne operacije na S: *množenje* $\cdot$ i *sabiranje* $+$, kao i dva elementa $0, 1 \in S$, takva da važi:

(i) $(S, +, 0)$ je komutativan monoid;

(ii) $(S, \cdot, 1)$ je monoid;

(iii) množenje i sabiranje zadovoljavaju *levi* i *desni distributivni zakon*, odnosno, za svako $x, y, z \in S$ važi:

$$x \cdot (y + z) = x \cdot y + x \cdot z,$$
$$(x + y) \cdot z = x \cdot z + y \cdot z;$$

(iv) Nula je *apsorbujući element*, odnosno važi:

$$x \cdot 0 = 0 \cdot x = 0, \quad \text{za svako } x \in S.$$

Za poluprsten S kaže se da *bez delioca nule* ako $x \cdot y = 0$ povlači $x = 0$ ili $y = 0$, za svako $x, y \in S$. Takođe, za S se kaže da je *komutativan* ako je i $(S, \cdot, 1)$ komutativan monoid, a kaže se da je *aditivno idempotentan* ako je $(S, +, 0)$ idempotentan monoid. Iz distributivnih zakona (iii) sledi da je S aditivno idempotentan poluprsten ako i samo ako je $1 + 1 = 1$.

Primer 1.2. (i) Tipičan primer poluprstena je $(\mathbb{N}, +, \cdot, 0, 1)$, odnosno, skup prirodnih brojeva sa uobičajenim operacijama sabiranja i množenja prirodnih brojeva. Takođe, skupovi celih brojeva $\mathbb{Z}$, racionalnih brojeva $\mathbb{Q}$, realnih brojeva $\mathbb{R}$ i kompleksnih brojeva $\mathbb{C}$, snabdeveni uobičajenim operacijama sabiranja i množenja, predstavljaju primere poluprstenova;

(ii) Vrlo važan primer poluprstena je *Boolov poluprsten* $(B, \max, \min, 0, 1)$ sa nosač skupom $B = \{0, 1\}$;

(iii) Neka je $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geq 0\}$. Tada su $(\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$, $(\mathbb{R}_+, +, \cdot, 0, 1)$ i $(\mathbb{R}_+ \cup \{\infty\}, +, \cdot, 0, 1)$ poluprsteni, usvjajaući dogovor $0 \cdot \infty = \infty \cdot 0 = 0$;

(iv) Poluprsteni $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ i $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ nazivaju se *tropski poluprsteni* ili *min-plus poluprsteni*;

(v) Poluprsteni $(\mathbb{N} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$ i $(\mathbb{R}_+ \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$ nazivaju se *arktičkim poluprsteni* ili *max-plus poluprsteni*, usvajajući dogovor $(-\infty) + \infty = -\infty$;

(vi) *Viterbiev poluprsten* predstavlja petorku $([0, 1], \max, \cdot, 0, 1)$, koja se sastoji od jediničnog realnog intervala sa operacijama max i uobičajenog množenja realnih brojeva;

(vii) Neka je X konačan alfabet. Tada je petorka $(2^{X^*}, \cup, \cdot, \emptyset, \{\varepsilon\})$, koja se sastoji od svih podskupova skupa X^* (koji se nazivaju *formalni jezici*), gde je sabiranje definisano kao unija skupova, a množenje sa $L_1 \cdot L_2 = \{\omega_1 \cdot \omega_2 | \omega_1 \in L_1 \text{ i } \omega_2 \in L_2\}$, predstavlja poluprsten. Ova petorka naziva se *poluprsten formalnih jezika nad alfabetom* X;

(viii) Neka je A neprazan skup. Tada je struktura $(2^{A \times A}, \cup, \circ, \emptyset, \triangle)$ poluprsten;

(ix) *Łukasiewiczev poluprsten* je struktura $([0, 1], \max, \otimes, 0, 1)$, gde je $x \otimes y = \max\{0, x + y - 1\}$, za svako $x, y \in [0, 1]$;

(x) Ako je L ograničena distributivna mreža (videti stranu 14), tada su strukture $(L, \vee, \wedge, 0, 1)$ i $(L, \wedge, \vee, 1, 0)$ poluprsteni;

(xi) *Max-min poluprsten* $R_{\max,\min} = (\mathbb{R}_+ \cup \{\infty\}, \max, \min, 0, 1)$, koji se sastoji od nenegativnih realnih brojeva sa operacijama max i min, koristi se u operacionim istraživanjima za problem maksimalnog kapaciteta mreža;

(xii) *Fazi poluprsten* $F = ([0, 1], \max, \min, 0, 1)$ često se koristi u teoriji fazi skupova.

Poluprsteni (i)-(vi), (ix) i (x) su komutativni, poluprsteni (ii), (iv)-(x) su aditivno idempotentni, dok svi poluprsteni osim (ix) nemaju delioca nule.

U sličnom maniru kao za monoide, za poluprsten S kaže se da je *lokalno konačan* ako je svaki konačno generisan podpoluprsten konačan. Ekvivalentno, poluprsten S je lokalno konačan ako su oba monoida $(S, +, 0)$ i $(S, \cdot, 1)$ lokalno konačna (videti [60, Poglavlje 3] i [58]). Na primer, ako su obe operacije sabiranja i množenja komutativne i idempotentne, tada je poluprsten lokalno konačan. Max-min, fazi, Łukasiewiczev poluprsten, kao i svaka ograničena distributivna mreža, primeri su lokalno konačnih poluprstena.

Mreže mogu biti okarakterisane i kao algebarske strukture koje zadovoljavaju određena svojstva. Preciznije, svaka komutativna i idempotentna polugrupa naziva se *polumreža*, dok se svaki komutativan i idempotentan monoid naziva *ograničena polumreža*. Tada je *mreža* algebarska struktura $(L, \wedge, \vee)$ takva da važi:

(1) $(L, \wedge)$ je polumreža, koja se nazivam *infimum-polumreža*, dok se binarna operacija $\wedge$ na L naziva *infimum*,

(2) $(L, \vee)$ je polumreža koja se naziva *supremum-polumreža*, dok se binarna operacija $\vee$ na L naziva *supremum*,

(3) Za svako $x, y \in L$ važe *zakoni apsopcije*:

$$x \vee (x \wedge y) = x \quad \text{i} \quad x \wedge (x \vee y) = x.$$

Dva zakona apsorpcije su jedine dve aksiome u kojima se u isto vreme pojavljuju infimum i supremum, i one obezbeđuju da se mreža razlikuje od proizvoljnog para polumreža.

Dve definicije mreža, preko uređenih skupova i algebarska definicija, međusobno su ekvivalentne. Zaista, ukoliko je $(L, \leq)$ mreža u smislu uređenih skupova, tada je $(L, \wedge, \vee)$ mreža kao algebarska struktura, pri čemu je $x \wedge y = \inf\{x, y\}$ i $x \vee y = \sup\{x, y\}$ za svako $x, y \in L$. Obratno, za datu mrežu $(L, \wedge, \vee)$ kao algebarsku strukturu, definiše se delimično uređenje $\leq$ na L sa $x \leq y$ ako i samo ako je $x \wedge y = x$ (ili ekvivalentno, ako i samo ako je $x \vee y = y$), za svako $x, y \in L$. Tada je $(L, \leq)$ mreža u kojoj je $\inf\{x, y\} = x \wedge y$ i $\sup\{x, y\} = x \vee y$, za svako $x, y \in L$. Štaviše, ove dve konstrukcije su međusobno inverzne. Stoga se obe definicije kroz nastavak izlaganja podjednako koriste.

Takođe, ako je $(L, \leq, 0, 1)$ ograničena mreža, tada je infimum-polumreža $(L, \wedge, 1)$ ograničena polumreža u kojoj je 1 najveći element, dok je supremum-polumreža $(L, \vee, 0)$ ograničena polumreža u kojoj je 0 najmanji element. Ograničenu mrežu označavamo takođe i sa $(L, \wedge, \vee, 0, 1)$.

Primer 1.3. (i) Trivijalan primer mreže prestavlja skup od jednog elementa $L = \{x\}$, i sa jedinim mogućim delimičnim uređenjem $x \leq x$. Ovu mrežu nazivamo *trivijalna mreža*;

(ii) Svaki lanac je mreža, ali ne nužno i kompletna. Na primer, $(\mathbb{Z}, \min, \max)$ (odnosno, skup celih brojeva sa prirodnim uređenjem) je mreža, iako ne kompletna. Skup $(\mathbb{N}, \min, \max)$ je mreža u kojoj je 1 najmanji element, a u kojoj ne postoji najveći element;

(iii) Na skupu prirodnih brojeva $\mathbb{N}$ definišimo delimično uređenje $\leq$ sa

$$x \leq y \quad \text{ako i samo ako} \quad x \text{ deli } y,$$

za svako $x, y \in \mathbb{N}$. Tada je $(\mathbb{N}, \text{nzd}, \text{nzs})$ mreža (pri čemu sa nzd označavamo najveći zajednički delilac, a sa nzs najmanji zajednički sadržalac dva prirodna broja). Primetimo da je 1 najmanji element ove mreže;

(iv) Za neprazan skup A, struktura $(2^A, \cap, \cup, \emptyset, A)$ predstavlja ograničenu mrežu (u kojoj je $\emptyset$ najmanji element, dok je A najveći element);

(v) Kolekcija svih particija nepraznog skupa A, uređena operacijom usitnjenja, predstavlja mrežu.

(vi) Struktura $(\mathbb{N} \times \mathbb{N}, \leq)$, sa delimičnim uređenjem $\leq$ definisanim sa

$$(x, y) \leq (z, v) \quad \text{ako} \quad x \leq z \text{ i } y \leq v,$$

za svako $x, y, z, v \in \mathbb{N}$, je mreža u kojoj je $(0, 0)$ najmanji element (a u kojoj ne postoji najveći element).

Distributivna mreža je mreža $(L, \wedge, \vee)$ koja zadovoljava jedan od sledeća dva ekvivalentna uslova za svako $x, y, z \in L$:

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$$

Dopunjena mreža je ograničena mreža $(L, \wedge, \vee, 0, 1)$ u kojoj za svaki element $x \in L$ postoji *dopuna*, odnosno, element $y \in L$ takav da je:

$$x \vee y = 1 \quad \text{i} \quad x \wedge y = 0.$$

Generalno, za proizvoljan element mreže može postojati više od jedne dopune. Mreža u kojoj za svaki element postoji tačno jedna dopuna naziva se *jednoznačno dopunjena mreža*. Primetimo da je svaka ograničena distributivna mreža jednoznačno dopunjena.

Boolova mreža je distributivna dopunjena mreža. Prema prethodnim zapažanjima, svaka Boolova mreža ujedno je i jednoznačno dopunjena. Sa x' označimo dopunu elementa x Boolove mreže. Tada važe sledeća svojstva.

Propozicija 1.4. *Neka je L Boolova mreža i neka su $x, y \in L$. Tada su sledeća svojstva zadovoljena:*

$$0' = 1, \quad 1' = 0 \quad i \quad x'' = x,$$
$$(x \wedge y)' = x' \vee y' \quad i \quad (x \vee y)' = x' \wedge y',$$
$$x \leq y \quad akko \quad x \wedge y' = 0 \quad akko \quad x' \vee y = 1 \quad akko \quad y' \leq x'.$$

1.3 Kompletne reziduirane mreže

U klasičnoj teoriji izračunljivosti, nedeterministički konačni automat (skraćeno NKA) predstavlja matematički model sistema koji se sastoji od konačnog skupa stanja A, skupa ulaznih simbola ili alfabeta (monoid reči X^*), skupa početnih stanja (koji je podskup od A), skupa završnih stanja (takođe podskup od A), kao i relacije prelaza (zadata kao podskup od $A \times X \times A$) koja omogućuje prelaz iz jednog stanja u drugo pod uticajem simbola ulaznog alfabeta. Intuitivno, NKA počinje sa radom iz nekog početnog stanja, i obrađuje niz ulaznih simbola, te za svaki simbol sa ulaza prelazi u jedno od mogućih stanja na osnovu trenutnog stanja i učitanog simbola. Na kraju, kada i poslednji simbol bude učitan od strane NKA-a, niz simbola se *prihvata* ili *odbacuje* u zavisnosti od toga da li se NKA nalazi u nekom od završnih stanja.

Fazi konačni automat (skraćeno FKA) predstavlja proširenje modela NKA. Preciznije, FKA se sastoji od konačnog skupa stanja A, ulaznog alfabeta X, ali u kojem su skup početnih stanja i skup završnih stanja dati kao *fazi podskupovi* od A, dok je relacija prelaza data kao *fazi podskup* od $A \times X \times A$. Pod fazi podskupom, podrazumevamo podskup u kojem elementi nemaju jasnu pripadnost, već pripadaju podskupu *u određenom stepenu*. Ovim se postiže efekat da FKA jednostavno ne prihvata ili odbacuje ulazni niz simbola, već da ga *prihvata u određenom stepenu*. Jasno je da se ovim proširenjem sa modela NKA na model FKA dobija mnogo veća fleksibilnost u modeliranju fenomena koji se sreću u realnom, fizičkom svetu, u kojima kriterijumi pripadnosti nekoj klasi nisu jasno i precizno definisani. Na primer, uz pomoć NKA-a, moguće je modelirati mašinu koja na ulazu prihvata reči na engleskom jeziku, a na izlazu odgovarajuće reči na srpskom jeziku. Uz pomoć FKA-a, moguće je modelirati mašinu koja će za datu ulaznu reč na engleskom jeziku, na izlazu dati reči na srpskom jeziku koje *u nekom stepenu odgovaraju* datoj ulaznoj reči. Taj stepen može zavisiti od konteksta reči u celoj rečenici, te rečenice u celom paragrafu, tog paragrafa u celom teksu, itd.

Dakle, glavna ideja FKA-a je *gradacijsko prihvatanje reči*. Stoga, može se reći da je rad NKA-a "bivalentan", odnosno, da NKA prihvata reč u stepenu 1 (da je u potpunosti prihvata) ili u stepenu 0 (da je u potpunosti odbacuje). Sa druge strane, kod FKA-a, svakoj reči dodeljuje se stepen iz određene skale L stepena istinitosti, tako da taj stepen odgovara stepenu prihvatanja te reči od strane datog FKA-a. S obzirom da je potrebno modelirati gradacijsko prihvatanje reči, jasno je da je potrebno da u skali L postoji delimično uređenje $\leq$ u kojem je 0 najmanji element i 1 najveći element. Time se postiže efekat da se neke reči *u većem stepenu* prihvataju od drugih reči, kao i

da se reči sa stepenom istinitosti 1 *u potpunosti* (*jasno*, *precizno*) *prihvataju*, dok se reči sa stepenom istinitosti 0 *u potpunosti* (*jasno*, *precizno*) *odbacuju*. Na taj način, NKA postaje posebna instanca FKA u kojima je skala istinitosti jednaka $L = \{0, 1\}$ i $0 \leq 1$. Na primer, za datu ulaznu reč "fuzzy" na engleskom jeziku, na izlazu FKA-a mogu se naći, recimo, reči "nejasan" u stepenu 0.9, "pomućen" u stepenu 0.7, "čupav" u stepenu 0 ako se reč fuzzy našla u konteksu *fuzzy logic*, dok su se na izlazu mogle naći reči, recimo, "nejasan" u stepenu 0.3, "pomućen" u stepenu 0.1, "čupav" u stepenu 1 ako se reč fuzzy našla u konteksu *fuzzy animal*.

Očigledno, najčešći i najpopularniji primer skale stepena istinitosti L je realan jedinični interval $[0, 1]$, ali u opštem slučaju, skala L ne mora biti linearno uređena.

Kod NKA može postojati više putanja koje prolaze kroz različita stanja pod uticajem niza ulaznih simbola, a reč se prihvata ukoliko postoji putanja koja počinje iz nekog stanja iz skupa početnih stanja, a završava se u nekom stanju iz skupa završnih stanja NKA-a. Kao i kod NKA-a, i kod FKA-a može postojati više putanja koje prolaze kroz različita stanja pod uticajem ulaznih simbola, s tim što je svakom prelazu dodeljen odgovarajući stepen iz skale L. Da bi se odgovorilo na pitanje u kom stepenu se prihvata reč kod datog FKA-a, moramo biti u stanju da modeliramo "postojanje putanje koja počinje iz nekog stanja iz skupa početnih stanja, a završava se u nekom stanju iz skupa završnih stanja FKA-a". Ukoliko se svakoj putanji u FKA-u dodeli neki stepen istinitosti, tada se postojanje goreopisane putanje može modelirati supremumom stepena svih takvih putanja u FKA-u. Intuitivno, određivanje postojanja putanje koja počinje iz nekog stanja iz skupa početnih stanja, a završava se u nekom stanju iz skupa završnih stanja FKA-a ekvivalentno je prolazu kroz sve takve putanje i nalaženju "najbolje" takve putanje (odnosno, pronalaženju najmanje vrednosti veće ili jednake od stepena istinitosti svih putanja), što je upravo šta supremum radi. Slično, može se tražiti i postojanje infimuma u skali L radi modeliranja "za sve određene putanje u FKA-u". Dakle, u fazi logici, supremum služi za modeliranje egzistencijalnog kvantifikatora, dok infimum služi za modeliranje univerzalnog kvantifikatora. Stoga se zahteva da skala L bude mreža u kojoj je 0 najmanji element, a 1 najveći element, odnosno, da $(L, \wedge, \vee, 0, 1)$ bude ograničena mreža.

Ostalo je odgovoriti na pitanje i kako dodeliti stepen istinitosti nekoj putanji u FKA-u koja prolazi kroz neka stanja pod uticajem ulaznih simbola. Intuitivno, potrebno je uvesti neku operaciju "množenja" na L, takvu da je moguće zapravo pomnožiti stepene istinitosti svih prelaza u nekoj putanji, i tu vrednost dodeliti toj putanji. Očigledno, operacija množenja, koju označavamo sa $\otimes$, biće binarna operacija na L koja zadovoljava određena svojstva. Neka su a, b, c i d stanja FKA-a takva da postoje prelazi između a i b, b i c, kao i c i d u nekim stepenima, koje ćemo označiti sa x, y i z, respektivno. Tražimo da važi $x \otimes y = y \otimes x$, odnosno, da bismo odredili stepen istinitosti putanje između a i c, svejedno je u kom redosledu se množe stepeni istinitosti prelaza između stanja a i b i prelaza između stanja b i c. Takođe, zahtevamo i da važi $x \otimes (y \otimes z) = (x \otimes y) \otimes z$, odnosno, redosled množenja takođe nije bitan. Na kraju, tražimo da je $x \otimes 1 = x$, za svako $x \in L$, odnosno, da pojedinačni prelazi sa stepenom istinitosti 1 ne utiču na konačan stepen istinitosti cele putanje. Svi uslovi vode do toga da struktura $(L, \otimes, 1)$ bude komutativan monoid. Osim toga, poželjno je i da operacija $\otimes$ bude neopadajuća, ili formalno rečeno, za $x_1 \leq x_2$ i $y_1 \leq y_2$ važi $x_1 \otimes y_1 \leq x_2 \otimes y_2$, za svako $x_1, x_2, y_1, y_2 \in L$. Ovaj uslov nam omogućuje da veći stepeni pojedinačnih prelaza rezultuju većim stepenom cele putanje. Kao što je pokazano u Sekciji 1.4, kada je $L = [0, 1]$ jedinični interval na realnim brojevima, tada se operacija $\otimes$ modelira takozvanim *t-normama*, no L može biti i mnogo opštija struktura.

Na kraju, zahtevaćemo da važi još jedan uslov, a njegovu egzistenciju objašnjavamo kroz sledeću situaciju. Neka su a, b i c stanja fazi automata takva da postoji prelaz između stanja a i b sa stepenom $x \in L$, kao i putanja između stanja b i c koja može da prolazi kroz neka međustanja fazi automata, i neka je težina te putanje jednaka $y \in L$. S obzirom da postoji više mogućnosti da se iz stanja b dostigne stanje c, a L ograničena mreža, tada je zapravo y jednako supremumu težina svih takvih mogućnosti (s obzirom da supremum modelira egzistencijalnog kvantifikatora, kako je malopre objašnjeno). Preciznije, imamo da je $y = \bigvee_{i \in I} y_i$, pri čemu je y_i težina pojedinačne mogućnosti prelaza iz b u c. S obzirom da $y = \bigvee_{i \in I} y_i$ modelira postojanje putanje od stanja b do stanja c, a x je težina prelaza od stanja a do stanja b, zahtevaćemo da $x \otimes y = x \otimes (\bigvee_{i \in I} y_i)$ modelira postojanje putanje od stanja a do stanja c. Formalno rečeno, za svako $x \in L$ i $y_i \in L$, za svako $i \in I$ iz indeksnog skupa I zahtevamo da važi:

$$\bigvee_{i \in I} (x \otimes y_i) = x \otimes \bigvee_{i \in I} y_i. \tag{1.10}$$

Kao što će biti pokazano (videti Teoremu 1.11), uslov (1.10) ekvivalentan je mnogim drugim uslovima, a između ostalih, ekvivalentan je i uslovu postojanja neke vrste *reverzne* operacije operaciji $\otimes$, odnosno postojanju vrste operacije *deljenja* ili *reziduala* na skali L, a koju označavamo sa $\rightarrow$, i za koju važi $x \otimes y \leq z$ ako i samo ako važi $x \leq y \rightarrow z$, za svako $x, y, z \in L$. Obe operacije $\otimes$ i $\rightarrow$ koriste se kako u formulaciji pravila zaključivanja u fazi logici, kao i u teoriji fazi skupova i fazi relacija. Operacija $\rightarrow$ neophodna je za rešavanje sistema nejednačina u L, što će nam biti neophodno u dva slučaja: za računanje relacija nad skupom stanja automata za modeliranje identičnih stanja, kao i u algoritmima za konstrukciju determinističkih fazi automata. U fazi logici, $\otimes$ modelira konjukciju, $\rightarrow$ modelira implikaciju, dok uslov $x \otimes y \leq z$ akko $x \leq y \rightarrow z$ modelira *modus ponens* pravilo zaključivanja (za više detalja, videti Bělohlávek [15], Bělohlávek i Vychodil [16], Ignjatović [94]).

Algebarska struktura koja zadovoljava sve prethodno navedene uslove naziva se *reziduirana mreža*. Kao vrlo opšta algebarska strutkura koja zadovoljava mnonogbrojna svojstva, našla se u fokusu istaživanja algebrista već krajem 1930.-ih, dok se danas intenzivno izučava i od strane inženjera i drugih istraživača iz primenjenih grana nauke, s obzirom da se koristi kako u fazi logici, tako i u drugim neklasičnim logikama (videti [15, 66, 86, 89]). Takođe, rezidurane mreže koriste se i u veštačkoj inteligenciji u nekim modelima rasuđivanja [170].

Definicija 1.5. *Reziduirana mreža* je algebarska struktura $\mathcal{L} = (L, \wedge, \vee, \otimes, \rightarrow, 0, 1)$, koja se sastoji od nepraznog skupa L, četiri binarne operacije $\wedge, \vee, \otimes, \rightarrow$ na L, i elemenata $0, 1 \in L$ tako da važi:

(1) $(L, \wedge, \vee, 0, 1)$ je ograničena mreža;

(2) $(L, \otimes, 1)$ je komutativan monoid;

(3) Za sve $x, y, z \in L$, važi *svojstvo adjunkcije*:

$$x \otimes y \leq z \quad \text{akko} \quad x \leq y \rightarrow z. \tag{1.11}$$

Ako je, dodatno, $(L, \wedge, \vee, 0, 1)$ kompletna ograničena mreža, tada se $\mathcal{L}$ naziva *kompletna reziduirana mreža*.

Da bi se naglasila monoidna struktura, u nekim izvorima (na primer, [20, 89, 90]) reziduirane mreže nazivaju se *inegralni, komutativni, reziduirani l-monoidi*, dok je struktura $(L, \leq, \otimes)$ izučavana i pod imenom *Lindenbaum algebra* [90].

Operacije $\otimes$ i $\rightarrow$ nazivaju se, redom, *množenje* i *rezidual* (ili *ostatak*), dok se uređeni par $(\otimes, \rightarrow)$ naziva *adjungovani par*. Za $x, y \in L$, $x \rightarrow y$ naziva se ostatak od y sa

x. Za datu operaciju $\otimes$ jednoznačno je određena operacija $\to$, i obratno. Kao što je ranije napomenuto, operacije množenja $\otimes$ i reziduala $\to$ namenjene su za modelovanje konjukcije i implikacije u odgovarajućoj logici, dok su supremum $\bigvee$ i infimum $\bigwedge$ namenjeni za modelovanje univerzalnog i egzistencijalnog kvantifikatora, respektivno.

Primer 1.6. Neka je $L = [0,1]$ jedinični interval u skupu realnih brojeva sa prirodnim delimičnim uređenjem na njemu. Tada svaki adjungovani par $(\otimes, \to)$, dat preko formula (1.12)–(1.14), čini algebru $\mathcal{L} = (L, \min, \max, \otimes, \to, 0, 1)$ kompletnom reziduiranom mrežom:

$$x \otimes y = \max\{x + y - 1, 0\}, \quad x \to y = \min\{1 - x + y, 1\}, \tag{1.12}$$

$$x \otimes y = \min\{x, y\}, \quad x \to y = \begin{cases} 1 & x \leq y, \\ y & \text{inače}, \end{cases} \tag{1.13}$$

$$x \otimes y = x \cdot y, \quad x \to y = \begin{cases} 1 & x \leq y, \\ y/x & \text{inače}. \end{cases} \tag{1.14}$$

Algebra $\mathcal{L}$ naziva se *standardna Łukasiewiczeva algebra* kada je adjungovani par definisan sa (1.12), *standardna Gödelova algebra* kada je adjungovani par definisan sa (1.13), i *standardna Gougenova* (ili *proizvod*) *algebra* kada je adjungovani par definisan sa (1.14).

Primer 1.7. Neka je $L = \{x_0, x_1, \ldots, x_n\}$ sa delimičnim uređenjem $0 = x_0 < x_1 < \ldots < x_n = 1$. Definišimo infimum i suprermum sa $x_i \wedge x_j = x_{\min\{i,j\}}$ i $x_i \vee x_j = x_{\max\{i,j\}}$, za svako $i, j \in \{0, 1, \ldots, n\}$. Tada svaki adjungovani par $(\otimes, \to)$ zadat sa (1.15) i (1.16) čini algebru $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, x_0 = 0, x_n = 1)$ kompletnom reziduiranom mrežom:

$$x_k \otimes x_l = x_{\max\{k+l-n,0\}}, \quad x_k \to x_l = x_{\min\{n-k+l,n\}}, \tag{1.15}$$

$$x_k \otimes x_l = x_{\min\{k,l\}}, \quad x_k \to x_l = \begin{cases} 1 & k \leq l, \\ x_l & \text{inače}, \end{cases} \tag{1.16}$$

U slučaju kada je $\{x_0, x_1, \ldots, x_n\} \subseteq [0,1]$ i $x_i = i/n$, tada su operacijue $(\otimes, \to)$ definisane sa (1.15) restrikcije operacija (1.12) od $[0,1]$ na $\{x_0, x_1, \ldots, x_n\}$, tj. $\{x_0, x_1, \ldots, x_n\}$ je poduniverzum standardne Łukasiewiczeve algebre. Slično, kada je $\{x_0, x_1, \ldots, x_n\} \subseteq [0,1]$, $x_0 = 0$ i $x_n = 1$, tada su operacije $(\otimes, \to)$ definisane sa (1.16) restrikcije operacija (1.13) od $[0,1]$ na $\{x_0, x_1, \ldots, x_n\}$, tj. $\{x_0, x_1, \ldots, x_n\}$ je poduniverzum standardne Gödelove algebre.

U slučaju kada je $L = \{0, 1\}$, tada obe strukture iz Primera 1.7 rezultiraju dvoelementnom Boolovom mrežom, odnosno, strukturom klasične dvovrednosne logike. Takođe, ne postoji druga reziduirana mreža na skupu $\{0, 1\}$.

Sledeća svojstva su zadovoljena u svakoj kompletnoj reziduiranoj mreži. Za dokaz videti [15, 16].

Propozicija 1.8. *Za svako $x, y \in L$, sledeća svojstva važe u svakoj kompletnoj reziduiranoj mreži $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$:*

$$x \otimes (x \to y) \leq y, \quad y \leq x \to (x \otimes y), \quad x \leq (x \to y) \to y, \tag{1.17}$$

$$x \otimes y \leq x, \quad x \leq y \to x, \tag{1.18}$$

$$x \leq y \quad akko \quad x \to y = 1, \tag{1.19}$$
$$x \to x = 1, \tag{1.20}$$
$$1 \to x = x, \tag{1.21}$$
$$x \otimes 0 = 0, \quad x \to 1 = 1, \quad 0 \to x = 1, \tag{1.22}$$
$$(x \to y) \otimes (z \to w) \leq (x \otimes z) \to (y \otimes w), \tag{1.23}$$
$$x \otimes (x \to y) = y \qquad akko \qquad (\exists z)\, x \otimes z = y, \tag{1.24}$$
$$x \to y \text{ je najveći element skupa } \{z | x \otimes z \leq y\}, \tag{1.25}$$
$$x \otimes y \text{ je najmanji element skupa } \{z | x \leq y \to z\}. \tag{1.26}$$

Propozicija 1.9. *U kompletnoj reziduiranoj mreži $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$, operacija $\otimes$ je rastuća po oba argumenta, dok je $\to$ rastuća po drugom, a opadajuća po prvom argumentu. Drugim rečima, za sve $x, x_1, x_2, y, y_1, y_2 \in L$, važi:*

$$x_1 \leq x_2 \; i \; y_1 \leq y_2 \quad povlači \quad x_1 \otimes y_1 \leq x_2 \otimes y_2, \tag{1.27}$$
$$x_1 \leq x_2 \quad povlači \quad x_2 \to y \leq x_1 \to y, \tag{1.28}$$
$$y_1 \leq y_2 \quad povlači \quad x \to y_1 \leq x \to y_2. \tag{1.29}$$

Propozicija 1.10. *Sledeća svojstva važe za svako $\{x_i\}_{i \in I} \subseteq L$ i $\{y_i\}_{i \in I} \subseteq L$ u svakoj kompletnoj reziduiranoj mreži $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$:*

$$x \otimes \bigvee_{i \in I} y_i = \bigvee_{i \in I} (x \otimes y_i), \tag{1.30}$$
$$x \otimes \bigwedge_{i \in I} y_i \leq \bigwedge_{i \in I} (x \otimes y_i), \tag{1.31}$$
$$x \to \bigwedge_{i \in I} y_i = \bigwedge_{i \in I} (x \to y_i), \tag{1.32}$$
$$\bigvee_{i \in I} x_i \to y = \bigwedge_{i \in I} (x_i \to y), \tag{1.33}$$
$$\bigvee_{i \in I} (x \to y_i) \leq x \to \bigvee_{i \in I} y_i, \tag{1.34}$$
$$\bigvee_{i \in I} (x_i \to y) \leq \bigwedge_{i \in I} x_i \to y. \tag{1.35}$$

Naredna Teorema pokazuje da se svojstvo adjunkcije može predstaviti na više ekvivalentnih načina (videti [15, 16]).

Teorema 1.11. *Neka je $(L, \wedge, \vee, \otimes, 0, 1)$ struktura koja zadovoljava uslove **(1)** i **(2)** iz Definicije 1.5. Tada su sledeći uslovi ekvivalentni:*

1. *Postoji binarna operacija $\to$ na L takva da par $(\otimes, \to)$ zadovoljava svojstvo adjunkcije* (1.11)*,*
2. *Za svako $x, y \in L$, skup $\{z | x \otimes z \leq y\}$ ima najveći element,*
3. *$x \otimes \bigvee_{i \in I} y_i = \bigvee_{i \in I} (x \otimes y_i)$ važi za svako $x \in L$ i $y_i \in L$, $i \in I$,*
4. *Za binarnu operaciju $\to$ na L definisanu sa $x \to y = \bigvee \{z | x \otimes z \leq y\}$, par $(\otimes, \to)$ zadovoljava svojstvo adjunkcije* (1.11)*.*

Neka je $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$ reziduirana mreža. Tada je *birezidual* binarna operacija $\leftrightarrow$ na L definisana sa:

$$x \leftrightarrow y = (x \to y) \wedge (y \to x),$$

za svako $x, y \in L$, dok je *negacija* $\neg$ unarna operacija na L definisana sa:

$$\neg x = x \to 0,$$

za svako $x \in X$. Operacija birezidуala $\leftrightarrow$ namenjena je za modelovanje jednakosti istinitosnih vrednosti, dok je negacija $\neg$ namenjena za modelovanje dopune (komplementa) za datu istinitosnu vrednost u odgovarajućim logikama. Na kraju, za svako $x \in L$ i $n \in \mathbb{N}_0$ definiše se *n-ti stepen* od x rekurzivno sa:

$$x^0 = 1 \quad \text{i} \quad x^{n+1} = x^n \otimes x.$$

Primer 1.12. Birezidual, negacija i n-ti stepen su, respektivno, definisani za svako $x, y \in L$ i $n \in \mathbb{N}_0$ sa:

- U standardnoj Łukasiewiczevoj algebri:

$$x \leftrightarrow y = 1 - |x - y|, \quad \neg x = 1 - x, \quad x^n = \max\{0, 1 - n(1 - x)\},$$

- U standardnoj Gödelovoj algebri:

$$x \leftrightarrow y = \begin{cases} 1, & x = y \\ \min\{x, y\}, & \text{inače} \end{cases}, \quad \neg x = \begin{cases} 1, & x = 0 \\ 0, & \text{inače} \end{cases}, \quad x^n = x,$$

- U standardnoj proizvod algebri:

$$x \leftrightarrow y = \frac{\min\{x, y\}}{\max\{x, y\}},$$

  negacija kao u standardnoj Gödelovoj algebri, a n-ti stepen kao uobičajeni n-ti stepen od x u skupu realnih brojeva.

Naredna teorema pokazuje neka osnovna svojstva birezidуma.

Propozicija 1.13. *U svakoj kompletnoj reziduiranoj mreži* $\mathcal{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$ *važi:*

$$\begin{aligned}
&0 \leftrightarrow 1 = 1 \leftrightarrow 0 = 0, \quad 0 \leftrightarrow 0 = 1 \leftrightarrow 1 = 1,\\
&x \leftrightarrow x = 1,\\
&x \leftrightarrow y = y \leftrightarrow x,\\
&(x \leftrightarrow y) \otimes (y \leftrightarrow z) \le x \leftrightarrow z,\\
&x \leftrightarrow 1 = x, \quad x \leftrightarrow 0 = \neg x,\\
&x \leftrightarrow y = (x \vee y) \to (x \wedge y),\\
&\bigwedge_{i \in I} (x_i \leftrightarrow y_i) \le \left(\bigwedge_{i \in I} x_i \right) \leftrightarrow \left(\bigwedge_{i \in I} y_i \right),\\
&\bigwedge_{i \in I} (x_i \leftrightarrow y_i) \le \left(\bigvee_{i \in I} x_i \right) \leftrightarrow \left(\bigvee_{i \in I} y_i \right).
\end{aligned}$$

Navodimo neke dodatne uslove za reziduirane mreže koje vode do posebnih istinitosnih struktura, a koje su od posebnog interesa za dalje izlaganje.

BL-algebra (*Basic Logic Algebra*) je reziduirana mreža $\mathcal{L}$ koja zadovoljava sledeća svojstva: *svojstvo deljivosti*:

$$x \wedge y = x \otimes (x \to y), \tag{1.36}$$

kao i *svojstvo prelinearnosti*:

$$(x \to y) \vee (y \to x) = 1, \tag{1.37}$$

za svako $x, y \in L$. BL-algebra koriste se u okruženju za interpretaciju formula Hájekove osnovne logike [86, 79]. Napominjemo da je svojstvo deljivosti ekvivalentno sa:

$$x \leq y \quad \text{povlači} \quad (\exists z \in L)\, x = y \otimes z, \tag{1.38}$$

za svako $x, y \in L$ (videti [16]).

Postoji nekoliko važnih klasa BL-algebri. Naime, *MV-algebra* (*Multi Valued Algebra*) je BL-algebra koja zadovoljava *zakon dvostruke negacije*:

$$\neg\neg x = x, \quad \text{za svako } x \in L.$$

Π-*algebra* (*Product algebra*) je BL-algebra koja zadovoljava sledeća dva svojstva:

$$(z \to 0) \to 0 \leq ((x \otimes z) \to (y \otimes z)) \to (x \to y),$$
$$x \wedge (x \to 0) = 0,$$

za svako $x, y, z \in L$. Na kraju, *G-algebra* (*Gödel algebra*) je BL-algebra koja zadovoljava *zakon idempotencije*:

$$x \otimes x = x, \quad \text{za svako } x \in L.$$

Razmotrimo strukture iz Primera 1.6 i 1.7. Tada je standardna Łukasiewiczeva algebra MV-algebra, standardna Gödelova algebra je G-algebra, dok je standardna proizvod algebra Π-algebra.

Heytingova algebra je reziduirana mreža $\mathcal{L}$ u kojoj za svako $x, y \in L$ važi:

$$x \otimes y = x \wedge y,$$

dok se reziduirana mreža koja je istovremeno Heytingova algebra i MV-algebra naziva *Boolova algebra*. Može se pokazati da je svaka konačna Boolova algebra izomorfna Boolovoj algebri $(2^A, \cap, \cup, \emptyset, A)$ svih podskupova konačnog skupa A. Definicije Boolove algebre i Boolove mreže međusobno su ekvivalentne. Zaista, ako je $(L, \wedge, \vee, 0, 1)$ Boolova mreža, tada je struktura $(L, \wedge, \vee, \otimes, \to, 0, 1)$ Boolova algebra, pri čemu je $x \otimes y = x \vee y$ i $x \to y = x' \wedge y$, za svako $x, y \in L$. Obratno, ako je $(L, \wedge, \vee, \otimes, \to, 0, 1)$ Boolova algebra, tada je $(L, \wedge, \vee, 0, 1)$ Boolova mreža u kojoj je operacija dopune $'$ definisana sa $x' = x \to 0$ za svako $x \in L$.

Neka je $\mathcal{L}$ kompletna reziduirana mreža. Ukoliko se izostave operacije infimum i rezidual, dobija se algebra $\mathcal{L}^* = (L, \vee, \otimes, 0, 1)$ koja je komutativan poluprsten. Preciznije, $(L, \vee, 0)$ i $(L, \otimes, 1)$ su komutativni monoidi, $\otimes$ je distributivna nad $\vee$, i $0 \otimes x = 0$, za svako $x \in L$. Struktura $\mathcal{L}^*$ naziva se *poluprstenski ostatak* od $\mathcal{L}$. S obzirom da je $(L, \vee, 0)$ polumreža, a svaka polumreža je lokalno konačna, imamo da je $\mathcal{L}^*$ lokalno konačan ako i samo ako je monoid $(L, \otimes, 1)$ lokalno konačan.

1.4 Reziduirane mreže i t-norme

Reziduirane mreže iz Primera 1.6 predstavljaju specijalne slučajeve reziduiranih mreža indukovanih takozvanim t-normama. *T-norma* (ili *trougaona norma*) kao binarna operacija uvedena je u radu Schweizer i Sklar [192]. Od tada, t-norme uglavnom su korišćene u probabilističkim metričkim prostorima i fazi logici, s obzirom da predstavljaju generalizaciju operacije preseka u mreži, kao i konjukcije u logici [86].

Kao što će uskoro biti pokazano, postoji korespodencija između reziduiranih mreža na realnom jediničnom intervalu $[0,1]$ i levo neprekidnih t-normi. Pored toga, postoji korespodencija između reziduiranih mreža definisanih na $[0,1]$ koje zadovoljavaju svojstvo deljivosti i neprekidnih t-normi.

T-norma je binarna operacija $*$ na jediničnom intervalu u skupu realnih brojeva $[0,1]$ tako da važi:

(1) $([0,1],*,1)$ je komutativan monoid;

(2) $*$ je kompatibilna sa prirodnim uređenjem na $[0,1]$, odnosno:

$$x \leq y \text{ i } z \leq v \text{ povlači } x * z \leq y * v, \quad \text{za svako } x,y,z,v \in [0,1].$$

T-norma je *levo-neprekidna* (odnosno, *desno-neprekidna*) ako za svako $y \in [0,1]$ i za svaki neopadajući (odnosno, nerastući) niz $\{x_n\}_{n\in\mathbb{N}} \in [0,1]^{\mathbb{N}}$ važi:

$$\lim_{n\to\infty}(x_n * y) = \left(\lim_{n\to\infty} x_n\right) * y,$$

dok je t-norma *neprekidna* ako je ujedno i levo- i desno-neprekidna, ili drugim rečima, ako za sve konvergentne nizove $\{x_n\}_{n\in\mathbb{N}}, \{y_n\}_{n\in\mathbb{N}} \in [0,1]^{\mathbb{N}}$ važi:

$$\lim_{n\to\infty}(x_n * y_n) = \left(\lim_{n\to\infty} x_n\right) * \left(\lim_{n\to\infty} y_n\right).$$

S obzirom da je jedinični interval $[0,1]^2$ kompaktan podskup realne ravni R^2, zaključujemo da je t-norma neprekidna (odnosno, levo-, odnosno, desno-neprekidna) akko je neprekidna (odnosno, levo-, odnosno, desno-neprekidna) kao realna funkcija dve promenljive, tj. u uobičajenoj topologiji na $[0,1]^2$.

Realna funkcija dve promenljive može biti neprekidna po svakoj promenljivoj a da ne bude neprekidna na skupu $[0,1]\times[0,1]$. Za proizvoljnu t-normu, naprotiv, ovo tvrđenje ne važi. Naime, sledeća lema važi za sve t-norme.

Lema 1.14. *T-norma $*$ je neprekidna akko je neprekidna u jednoj promenljivoj, tj. akko je funkcija $f_y(x) = x * y$ neprekidna za svako $y \in [0,1]$. Analogna tvrđenja važe za levu- i desnu-neprekidnost t-norme.*

Sledeće dve leme uspostavljaju korespodenciju između neprekidnih t-normi i reziduiranih mreža na $[0,1]$.

Lema 1.15. *T-norma $*$ je levo-neprekidna akko je struktura $([0,1],\min,\max,*,\Rightarrow,0,1)$ reziduirana mreža, pri čemu su* min *i* max *binarne operacije definisane na uobičajen način na $[0,1]$, a $\Rightarrow$ binarna operacija na $[0,1]$ definisana sa*

$$x \Rightarrow y = \bigvee\{z \in [0,1] | x * z \leq y\}, \tag{1.39}$$

za svako $x,y \in [0,1]$.

Lema 1.16. *T-norma $*$ je neprekidna akko je struktura $([0,1],\min,\max,*,\Rightarrow,0,1)$ reziduirana mreža koja zadovoljava svojstvo deljivosti (*(1.36) *ili* (1.38)*), pri čemu su* min *i* max *binarne operacije definisane na uobičajen način na $[0,1]$, a $\Rightarrow$ binarna operacija na $[0,1]$ definisana sa* (1.39).

S obzirom da je svojstvo prelinearnosti (1.37) uvek zadovoljeno u linearno uređenim reziduiranim mrežama, te stoga i u reziduiranim mrežama na $[0,1]$, sledeća Teorema sledi direktno iz Leme 1.16.

Teorema 1.17. *T-norma $*$ je neprekidna akko je struktura $([0,1], \min, \max, *, \Rightarrow, 0, 1)$ BL-algebra, pri čemu je $\Rightarrow$ definisana sa* (1.39).

Za dokaz prethodnih tvrđenja pogledati [15, 16], dok za dodatna svojstva neprekidnih t-normi i njihovih reprezentacija videti [119, 120, 121].

1.5 Fazi podskupovi i fazi relacije

Teoriju fazi skupova inicirao je L. A. Zadeh u svom sada čuvenom radu [219] iz 1965. godine. Za razliku od običnih skupova, u kojima je pripadnost elemenata jasno određena, elementi kod fazi skupova mogu pripadati njima *u određenom stepenu*. Na taj način, fazi skupovi služe za modelovanje klasa objekata koje se susreću u realnom fizičkom svetu, a u kojima ne postoje precizno definisani kriterijumi pripadnosti. Uvođenjem pojma fazi skupa, Zadehova ideja bila je da se premosti jaz između mentalne reprezentacije realnosti koja zahteva gradacijski pojam pripadnosti, i uobičajene bivalentne matematičke reprezentacije. Zaista, klasična logika previše je rigidna da bi bila u mogućnosti da modeluje pojmove iz fizičkog sveta opisanih prirodnim jezikom, kao što su visoka temperatura, mlad čovek, veliki grad, itd, a u kojima je apriorno prisutna nepreciznost i nejasnost.

U istom radu, Zadeh je uveo pojam fazi relacije koji je dalje razvijao u radu iz 1971. godine [220], gde je uveo i pojmove fazi ekvivalencije i fazi uređenja. Nakon toga, u velikom broju radova razmatrani su razni aspekti tih fazi relacija [39, 40, 55, 111, 158, 159], tako da je danas teorija binarnih fazi relacija jedna od najznačajnih oblasti u teoriji fazi skupova. Upravo zato što je u mogućnosti da modelira klase objekata iz relanog fizičkog sveta u kojima je prirodno prisutna nepreciznost, fazi relacije našle su primenu u mnogim poljima, kao što su lingvistika [49], psihologija [128], veštačka inteligencija [19], itd.

U nastavku, $\mathcal{L}$ označava kompletnu reziduiranu mrežu, dok A označava neprazan skup. *Fazi podskup od A nad $\mathcal{L}$*, ili jednostavno *fazi podskup od A*, je svako preslikavanje $\alpha : A \to L$. Za svaki fazi podskup α od A, $\alpha(a) \in L$ interpretira se kao stepen istinitosti od "a pripada α", za $a \in A$. Jasno je da je definicija fazi podskupa generalizacija karakteristične funkcije χ_A od A, ili drugim rečima, karakteristične funkcije običnih skupova poklapaju se sa fazi poskupovima u slučaju kada je $L = \{0,1\}$. Skup svih fazi poskupova od A označava se sa L^A.

Neka su $\alpha, \beta \in L^A$ dva fazi podskupa od A. *Jednakost* od α i β definiše se kao obična jednakost preslikavanja, tj.

$$\alpha = \beta \quad \text{akko} \quad \alpha(a) = \beta(a), \ \text{za svako } a \in A,$$

i *inkluzija* od α i β takođe se definiše pokoordinatno:

$$\alpha \leq \beta \quad \text{akko} \quad \alpha(a) \leq \beta(a), \ \text{za svako } a \in A.$$

Može se pokazati da je inkluzija fazi podskupova $\leq$ delimično uređenje, stoga se $\leq$ naziva *delimično uređenje* fazi podskupova. Sa $\emptyset$ označava se *prazan fazi podskup* od A definisan sa $\emptyset(a) = 0$ za svako $a \in A$, dok se sa λ označava *pun fazi podskup* od A definisan sa $\lambda(a) = 1$ za svako $a \in A$. Tada je struktura $(L^A, \wedge, \vee, \emptyset, \lambda)$ kompletna distributivna odraničena mreža u kojoj su infimum (presek) i supremum (unija) proizvoljne familije $\{\alpha_i\}_{i \in I}$ fazi podskupova od A definisani kao preslikavanja iz A u L sa:

$$\left(\bigwedge_{i \in I} \alpha_i\right)(a) = \bigwedge_{i \in I} \alpha_i(a), \quad \left(\bigvee_{i \in I} \alpha_i\right)(a) = \bigvee_{i \in I} \alpha_i(a),$$

respektivno. Takođe, ukoliko se operacije $\otimes$ i $\rightarrow$ prošire pokoordinatno na fazi podskupove od A, dobijamo da je struktura $\mathcal{L}^A = (L^A, \wedge, \vee, \otimes, \rightarrow, \emptyset, \lambda)$ kompletna reziduirana mreža. Stoga, svi identiteti i nejednakosti koje važe u $\mathcal{L}$ važe i u $\mathcal{L}^A$.

Fazi podskup α od A naziva se *krisp* ako je $\alpha(a) \in \{0, 1\}$, za svako $a \in A$. S obzirom da su krisp fazi podskupovi karakteristične funkcije običnih podskupova, krisp fazi podskupovi od A odgovaraju običnim podskupovima od A. Stoga, ne pravimo razliku između krisp fazi podskupova od A i odgovarajućih običnih podskupova od A.

Za fazi podskup $\alpha \in L^A$ kaže se da je *normalizovan* ako postoji $a \in A$ tako da je $\alpha(a) = 1$. Takođe, za fazi podskup $\alpha \in L^A$, njegova *visina* $\|\alpha\| \in L$ definisana je sa:

$$\|\alpha\| = \bigvee_{a \in A} \alpha(a).$$

Fazi relacija (ili preciznije, *binarna fazi relacija*) na skupu A je svako preslikavanje iz $A \times A$ u L, tj. svaki fazi poskup od $A \times A$, i jednakost, inkluzija, presek i unija fazi relacija definiše se kao za fazi podskupove. Skup svih fazi relacija na A označava se sa $L^{A \times A}$. Krisp fazi relacija je fazi relacija koja uzima vrednosti u skupu $L = \{0, 1\}$, i odgovara običnoj binarnoj relaciji na A. Takođe se koristi oznaka *prazne fazi relacije* $\emptyset$ definisane sa $\emptyset(a, b) = 0$ za svako $a, b \in A$. *Jedinična fazi relacija* $\triangle_A \in L^{A \times A}$ definisana je sa $\triangle_A(a, a) = 1$ za svako $a \in A$ i $\triangle_A(a, b) = 0$ za svako $a, b \in A$ za koje je $a \neq b$.

Za neprazak skup A i fazi relaciju $\varphi \in L^{A \times A}$, njen *inverz* je fazi relacija $\varphi^{-1} \in L^{A \times A}$ definisana sa

$$\varphi^{-1}(b, a) = \varphi(a, b),$$

za svako $a, b \in A$. Štaviše, za fazi relacije $\varphi, \phi \in L^{A \times A}$ i fazi podskupove $\alpha, \beta \in L^A$, definišu se *proizvodi* ili *kompozicije* $\varphi \circ \phi \in L^{A \times A}$, $\alpha \circ \varphi \in L^A$, $\varphi \circ \alpha \in L^A$ i $\alpha \circ \beta \in L$ sa:

$$(\varphi \circ \phi)(a, c) = \bigvee_{b \in A} \varphi(a, b) \otimes \phi(b, c), \quad \text{za svako } a, c \in A, \tag{1.40}$$

$$(\alpha \circ \varphi)(b) = \bigvee_{a \in A} \alpha(a) \otimes \varphi(a, b), \quad \text{za svako } b \in A, \tag{1.41}$$

$$(\varphi \circ \alpha)(a) = \bigvee_{b \in A} \varphi(a, b) \otimes \alpha(b), \quad \text{za svako } a \in A, \tag{1.42}$$

$$\alpha \circ \beta = \bigvee_{a \in A} \alpha(a) \otimes \beta(a). \tag{1.43}$$

Formule (1.40), (1.41), (1.42) i (1.43) predstavljaju generalizaciju formula (1.1), (1.2), (1.3) i (1.4), respektivno. Na kraju, *proizvod* $x \otimes \alpha \in L^A$ fazi podskupa $\alpha \in L^A$ i skalara $x \in L$ definisan je za svako $a \in A$ sa:

$$(x \otimes \alpha)(a) = x \otimes \alpha(a). \tag{1.44}$$

Kada je skup A konačan, fazi relacije na A interpretiraju se kao matrice, dok se fazi podskupovi od A interpretiraju kao vektori sa elementima u L. Tada se proizvod fazi relacija posmatra kao matrični proizvod, proizvod fazi podskupa i fazi relacije kao proizvod vektora i matrice (u odgovarajućem redosledu), proizvod dva fazi podskupa kao skalarni proizvod vektora, i proizvod skalara i fazi podskupa kao proizvod skalara i vektora.

Proizvod fazi relacija je asocijativna binarna operacija, odnosno:

$$(\varphi \circ \phi) \circ \psi = \varphi \circ (\phi \circ \psi), \tag{1.45}$$

za svako $\varphi, \phi, \psi \in L^{A\times A}$. Takođe, za svako $\varphi \in L^{A\times A}$ važi:

$$\triangle_A \circ \varphi = \varphi = \varphi \circ \triangle_A.$$

Štaviše, za svako $\varphi, \phi \in L^{A\times A}$ i $\alpha, \beta \in L^A$ važi:

$$(\alpha \circ \varphi) \circ \phi = \alpha \circ (\varphi \circ \phi), \tag{1.46}$$
$$(\alpha \circ \varphi) \circ \beta = \alpha \circ (\varphi \circ \beta), \tag{1.47}$$
$$(\varphi \circ \phi) \circ \alpha = \varphi \circ (\phi \circ \alpha). \tag{1.48}$$

Stoga se sve zagrade u (1.45) – (1.48) mogu izostaviti.

Neka je $\alpha \in L^A$ i $\varphi \in L^{A\times A}$. Kaže se za α da je *proširujuć* ili *vidljiv* u odnosu na φ ako je:

$$\alpha(a) \otimes \varphi(a, b) \leq \alpha(b),$$

za svako $a, b \in A$, kao i da je *neraspoznatljiv* ili *neprimetljiv* u odnosu na φ ako:

$$\alpha(a) \otimes \alpha(b) \leq \varphi(a, b),$$

za svako $a, b \in A$.

Za fazi relaciju φ na A kaže se da je:

(R) *refleksivna* (ili *fazi refleksivna*), ako $\varphi(a, a) = 1$ za svako $a \in A$;

(S) *simetrična* (ili *fazi simetrična*), ako $\varphi(a, b) = \varphi(b, a)$, za svako $a, b \in A$;

(T) *tranzitivna* (ili *fazi tranzitivna*), ako $\varphi(a, b) \otimes \varphi(b, c) \leq \varphi(a, c)$, za svako $a, b, c \in A$.

Iako se fazi refleksivnost, fazi simetričnost i fazi tranzitivnost mogu izraziti preko različitih definicija (videti [63]), uslovi (R), (S) i (T) jedni su od najčešće korišćenih s obzirom da generalizuju njihove odgovarajuće krisp uslove (videti stranu 8). Takođe, uslovi (R), (S) i (T) kažu da su logičke formule prvog reda koje izražavaju refleksivnost, simetričnost i tranzitivnost tačne u stepenu 1.

Za fazi relaciju $\varphi \in L^A$, fazi relacija φ^∞ na A definisana sa

$$\varphi^\infty = \bigvee_{n\in\mathbb{N}} \varphi^n,$$

je najmanja tranzitivna fazi relacija na A koja sadrži φ (najmanja u smislu standarnog uređenja fazi relacija), i naziva se *tranzitivno zatvorenje* od φ.

Fazi relacija na A koja je refleksivna i tranzitivna naziva se *fazi kvazi-uređenje*. U pojedinim referencama, fazi kvazi-uređenja nazivaju se *fazi preuređenja* (videti, na primer, [24, 65, 129]), no mi se opredeljujemo za gornji termin iz istog razloga kao i za krisp kvazi-uređenja. Za svako fazi-kvazi uređenje φ na A važi:

$$\varphi \circ \varphi = \varphi.$$

Skup svih fazi kvazi-uređenja na A označava se sa $\mathcal{Q}(A)$. Struktura $(\mathcal{Q}(A), \wedge, \vee)$ predstavlja kompletnu mrežu, u kojoj se presek $\wedge$ poklapa sa presekom fazi relacija, dok se $\vee$, u opštem slučaju, ne poklapa sa unijom fazi relacija, već je definisan za

proizvoljnu familiju $\{\varphi_i\}_{i\in I}$ fazi kvazi-uređenja na A kao:

$$\varphi = \left(\bigvee_{i\in I}\varphi_i\right)^{\infty} = \bigvee_{n\in\mathbb{N}}\left(\bigvee_{i\in I}\varphi_i\right)^{n}.$$

Simetrično fazi kvazi-uređenje na skupu naziva se *fazi ekvivalencija* (ili *operator neraspoznavanja* ili *fazi sličnost* u nekim izvorima [54, 204]). Skup svih fazi ekvivalencija na skupu A označava se sa $\mathcal{E}(A)$.

Neka je $\varphi \in L^{A\times A}$. Tada *$\varphi$-afterset* od a, $a \in A$, predstavlja fazi podskup $a\varphi \in L^A$ definisan sa:

$$a\varphi(b) = \varphi(a,b), \quad \text{za svako } b \in A,$$

dok je *φ-foreset* od a fazi podskup $\varphi a \in L^A$ definisan sa:

$$\varphi a(b) = \varphi(b,a), \quad \text{za svako } b \in A.$$

Skup svih φ-aftersetova označava se sa A/φ, dok se skup svih φ-foresetova označava sa $A\backslash\varphi$. U slučaju da je $\varphi \in \mathcal{E}(A)$, tada se za svako $a \in A$ definiše *fazi klasa ekvivalencije* (ili samo *klasa ekvivalencije*) od φ sa a kao fazi podskup $\varphi^a \in L^A$ sa

$$\varphi^a = a\varphi = \varphi a,$$

odnosno, fazi klasa ekvivalencije od φ sa a ekvivalentna je φ-aftersetu ili φ-foresetu od a. Drugim rečima, stepen u kojem neko $b \in A$ pripada nekoj klasi ekvivalencije od $\varphi \in \mathcal{E}(A)$ jednako je stepenu u kojem su a i b ekvivalentni (odnosno, u kom stepenu pripadaju fazi ekvivalenciji φ). Skup svih klasa ekvivalencije od φ označava se sa A/φ.

U Sekciji 1.1 pokazano je da krisp realcije ekvivalencije na nepraznom skupu A jednoznačno odgovaraju particijama skupa A. Sada pokazujemo da analogna korespodencija važi i za fazi relacije i takozvane fazi particije koje se uvode u nastavku. Najpre se daju sledeće karakterizacije fazi klasa ekvivalencija koje su dokazane u radu Ćirić i ostali [39].

Lema 1.18. *Fazi podskup $\alpha \in L^A$ je fazi klasa ekvivalencije neke fazi ekvivalencije $\varphi \in \mathcal{E}(A)$ akko je α normalizovan, proširujuć i neraspoznatljiv u odnosu na φ.*

Lema 1.19. *Neka je $\varphi \in \mathcal{E}(A)$. Tada za svako $a, b \in A$ važi:*

(i) $\varphi(a,b) = \|\varphi^a \otimes \varphi^b\|$,

(ii) $\varphi^a = \varphi^b$ *akko* $\varphi(a,b) = 1$.

Fazi particija je svaki skup (dakle, krisp skup) π fazi podskupova od A takav da je:

(1) Svako $\alpha \in \pi$ je normalizovan fazi podskup od A;

(2) Za svako $a \in A$ postoji $\alpha \in \pi$ tako da je $\alpha(a) = 1$;

(3) Za svako $\alpha, \beta \in \pi$ važi:

$$\bigvee_{a\in A}\alpha(a)\otimes\beta(a) \le \bigwedge_{b\in A}\alpha(b)\leftrightarrow\beta(b). \tag{1.49}$$

U slučaju $L = \{0,1\}$ koncept fazi particije poklapa sa konceptom particije skupa. Kao što je navedeno u [39], nejednakost (1.49) može da se tumači kao "stepen preklapanja α i β (leva strana nejednakosti) jednaka je stepenu jednakosti α i β (desna strana nejednakosti)", što generalizuje svojstvo (3) krisp particija sa strane 9: "ako α

i β nisu disjunktni (odnosno, imaju stepen preklapanja 1), tada je $\alpha = \beta$ (odnosno, imaju stepen jednakosti 1)".

Propozicija 1.20. *Neka je $\varphi \in \mathcal{E}(A)$, π fazi particija skupa A, i neka je $\varphi/\pi \in L^{A\times A}$ fazi relacija definisana sa:*

$$\varphi/\pi(a,b) = \alpha_a(b),$$

za svako $a, b \in A$, gde je $\alpha_a \in \pi$ fazi podskup od A takav da je $\alpha_a(a) = 1$. Tada važi sledeće:

(1) *A/φ je fazi particija od A;*

(2) *φ/π je fazi ekvivalencija na A;*

(3) *$\varphi = \varphi/(A/\varphi)$ i $\pi = \pi/(\varphi/\pi)$.*

Za više informacija o fazi particijama i fazi ekvivalencijama videti knjige Bělohlávek [14], Bělohlávek i Vychodil [16], kao i radove Ćirić i ostali [39], De Baets i Mesiar [11], Demirci [53], Klawonn [117], Klawonn i Kruse [118].

Za fazi kvazi-uređenje φ na A, fazi relacija ϵ_φ definisana sa $\epsilon_\varphi = \varphi \wedge \varphi^{-1}$ je fazi ekvivalencija na A, i naziva se *prirodna fazi ekvivalencija* od φ.

Za proizvoljan fazi podskup α od A, fazi relacije φ_α i φ^α na A, definisane sa:

$$\varphi_\alpha(a,b) = \alpha(a) \to \alpha(b), \quad \varphi^\alpha(a,b) = \alpha(b) \to \alpha(a),$$

za svako $a, b \in A$, predstavljaju fazi kvazi-uređenja na A. Posebno, ako je α normalizovan fazi podskup, tada je on afterset od φ_α i foreset od φ^α. Takođe, za proizvoljan fazi podskup α od A, fazi relacija ϵ_α definisana sa:

$$\epsilon_\alpha(a,b) = \alpha(a) \leftrightarrow \alpha(b),$$

za svako $a, b \in A$, predstavlja fazi ekvivalenciju na A.

Naredna Teorema daje važna svojstva fazi kvazi-uređenja i prirodnih fazi ekvivalencija.

Teorema 1.21. [197] *Neka je $\varphi \in \mathcal{Q}(A)$ i ϵ prirodna fazi ekvivalencija od φ. Tada važi:*

(a) *Za proizvoljno $a, b \in A$, sledeći uslovi su ekvivalentni:*
- **(1)** *$\epsilon(a,b) = 1$;*
- **(2)** *$\epsilon^a = \epsilon^b$;*
- **(3)** *$\varphi a = \varphi b$;*
- **(4)** *$a\varphi = b\varphi$.*

(b) *Preslikavanja $a\varphi \to \epsilon^a$ iz A/φ u A/ϵ, kao i $a\varphi \to \varphi a$ iz A/φ u $A\backslash\varphi$, predstavljaju bijektivna preslikavanja.*

Neka je $\varphi, \phi \in L^{A\times A}$. Tada je *desni rezidual* od ϕ sa φ fazi relacija $\varphi\backslash\phi \in L^{A\times A}$ definisana za svalo $a, b \in A$ sa:

$$(\varphi\backslash\phi)(a,b) = \bigwedge_{c\in A} \varphi(c,a) \to \phi(c,b), \tag{1.50}$$

dok je *levi rezidual* od ϕ sa φ fazi relacija $\phi/\varphi \in L^{A\times A}$ definisana za svako $a, b \in A$ sa:

$$(\phi/\varphi)(a,b) = \bigwedge_{c\in A} \varphi(b,c) \to \phi(a,c). \tag{1.51}$$

Tada važe sledeća *svojstva adjunkcije* za fazi relacije $\varphi, \phi, \psi \in L^{A\times A}$(videti [98]):

$$\varphi \circ \psi \le \phi \quad \text{akko} \quad \psi \le \varphi\backslash\phi, \tag{1.52}$$

$$\psi \circ \varphi \leq \phi \quad \text{akko} \quad \psi \leq \phi / \varphi. \tag{1.53}$$

Drugim rečima, za date $\varphi, \phi \in L^{A\times A}$, $\varphi\backslash\phi$ je najveće rešenje fazi relacijske nejednačine $\varphi \circ \psi \leq \phi$, pri čemu je $\psi \in L^{A\times A}$ nepoznata fazi relacija. Slično, ϕ/φ je najveće rešenje fazi relacijske nejednačine $\psi \circ \varphi \leq \phi$, pri čemu je $\psi \in L^{A\times A}$ nepoznata fazi relacija.

Takođe, neka je $\alpha, \beta \in L^A$. Tada je *desni rezidual* od β sa α fazi relacija $\alpha\backslash\beta \in L^{A\times A}$ definisana za svako $a, b \in A$ sa:

$$(\alpha\backslash\beta)(a, b) = \alpha(a) \to \beta(b), \tag{1.54}$$

dok je *levi rezidual* od β sa α fazi relacija $\beta/\alpha \in L^{A\times A}$ definisana za svako $a, b \in A$ sa:

$$(\beta/\alpha)(b, a) = \alpha(a) \to \beta(b). \tag{1.55}$$

Jasno je da je $\beta/\alpha = (\alpha\backslash\beta)^{-1}$. Neka je $\alpha, \beta \in L^A$ i $\varphi \in L^{A\times A}$. Tada važe sledeća *svojstva adjunkcije*:

$$\alpha \circ \varphi \leq \beta \quad \text{akko} \quad \varphi \leq \alpha\backslash\beta, \tag{1.56}$$
$$\varphi \circ \alpha \leq \beta \quad \text{akko} \quad \varphi \leq \beta/\alpha. \tag{1.57}$$

Drugim rečima, za date $\alpha, \beta \in L^A$, $\alpha\backslash\beta$ je najveće rešenje fazi relacijske nejednačine $\alpha \circ \varphi \leq \beta$, pri čemu je $\varphi \in L^{A\times A}$ nepoznata fazi relacija. Slično, β/α je najveće rešenje fazi relacijske nejednačine $\varphi \circ \alpha \leq \beta$, pri čemu je $\varphi \in L^{A\times A}$ nepoznata fazi relacija. Takođe, definišimo i *birezidual* od α i β kao fazi relaciju $\alpha|\beta \in L^{A\times A}$ sa $\alpha|\beta = (\alpha\backslash\beta) \wedge (\alpha/\beta)$, ili ekvivalentno, za svako $a, b \in A$:

$$(\alpha|\beta)(a, b) = \alpha(a) \leftrightarrow \beta(b). \tag{1.58}$$

Sledeća tvrđenja navode određene osobine fazi kvazi-uređenja i fazi ekvivalencija (videti [204, 67]) koja će biti korišćena u nastavku.

Lema 1.22. *Neka su $\varphi, \phi \in \mathcal{Q}(A)$ fazi kvazi-uređenja na A. Tada je fazi relacija $\varphi \wedge \phi$ takođe fazi kvazi-uređenje na A..*

Lema 1.23. *Neka su $\varphi, \phi \in \mathcal{E}(A)$ fazi ekvivalencije na A. Tada je fazi relacija $\varphi \wedge \phi$ takođe fazi ekvivalencija na A.*

Lema 1.24. *Neka su $\varphi, \phi \in \mathcal{Q}(A)$ i $\varphi \leq \phi$. Tada je $\varphi \circ \phi = \phi$.*

Lema 1.25. *Neka su $\varphi, \phi \in \mathcal{Q}(A)$ i $\varphi \leq \phi$. Tada je $\varphi \leq (\phi^a/\phi^a)$ za svako $a \in A$.*

Lema 1.26. *Neka su $\varphi, \phi \in \mathcal{E}(A)$ i $\varphi \leq \phi$. Tada je $\varphi \leq (\phi^a|\phi^a)$ za svako $a \in A$.*

Lema 1.27. *Neka je $\alpha \in L^A$. Tada su fazi relacije $\alpha\backslash\alpha$ i α/α fazi kvazi-uređenja na A, dok je fazi relacija $\alpha|\alpha$ fazi ekvivalencija na A.*

1.6 Uređeni poluprsteni, dioidi i matrice

Kao što je prikazano u Sekcijama 1.3 i 1.5, mreže predstavljaju osnovnu algebarsku strukturu prilikom izučavanja fazi skupova i fazi relacija, kao i njihovih primena. Glavni razlog leži u postojanju delimičnog uređenja u mrežama. Prethodnih godina, u fokusu istraživanja bili su i *delimično uređeni poluprsteni* kao opštije algebarske strukture, a u kojima je i dalje prisutno delimično uređenje [76, 77, 78, 88, 203].

Posebno značajna klasa delimično uređenih poluprstena čine *dioidi*, koji čine klasu poluprstena koji nisu prsteni. Kao što će biti pokazano, dioidi predstavljaju vezu između algebarskih struktura u fazi logici i opštih algebarskih struktura, i mnogi algebarski rezultati koji se nalaze u literaturi vezanoj za fazi logiku mogu se posmatrati kao *svojstva dioida* [77, 10].

Viševrednosne relacije nad poluprstenima nazivaju se *matrice*, i dok su one dobro izučavane od strane algebrista, retko su razmatrane kao uopštenja običnih binarnih relacija, na način kao što su izučavane fazi relacije. Postoje dva glavna razloga za to, a prvi je, kao što je već napomenuto, nedostatak delimičnog uređenja na polupstenima, i on se može rešiti izborom delimično uređenih polupstena kao strukture vrednosti matrica. Drugi razlog leži u tome što skup $\{0,1\}$, koji se sastoji od nule i jedinice poluprstena, ne mora da bude podpoluprsten, pa se matrice sa vrednostima u skupu $\{0,1\}$ ne mogu razmatrati kao klasične dvovrednosne relacije. Kao što će biti pokazano, ovi problemi mogu se rešiti tako da se vrednosti matrica uzimaju iz aditivno idempotentnih poluprstena, koje predstavljaju široku klasu ne samo dioida, već i samih poluprstena.

Neka je $(S,+,\cdot,0,1)$ poluprsten i R relacija na S. Tada je *R kompatibilna sa sabiranjem* ako $R(x,y)$ povlači $R(x+z,y+z)$, za svako $x,y,z \in S$. Ako postoji delimično uređenje $\leq$ u S koje je kompatibilno sa sabiranjem, tj. ako za svako $x,y,z \in S$ važi:

$$x \leq y \quad \text{povlači} \quad x+z \leq y+z,$$

tada se za S kaže da je *delimično uređeni poluprsten*.

Na proizvoljnom poluprstenu S moguće je definisati binarnu relaciju $\leq$ sa

$$x \leq y \quad \text{akko} \quad (\exists z)\, x+z=y, \tag{1.59}$$

za svako $x,y \in S$, koja je kvazi-uređenje na S, i koja se naziva *kanoničko kvazi-uređenje* na S. Kanoničko kvazi-uređenje je kompatibilno sa $+$, ali nije obavezno antisimetrično. Poluprsten u kojem je kanoničko kvazi-uređenje zapravo delimično kvazi-uređenje, odnosno poluprsten u kojem je kanoničko kvazi-uređenje antisimetrično, naziva se *kanoničko uređen poluprsten* ili *dioid*. Drugim rečima, dioid je poluprsten u kojem kanoničko kvazi-uređenje čini poluprsten delimično uređenim. U diodu, kanoničko kvazi-uređenje naziva se *kanoničko delimično uređenje*. Sledeće tvrđenje koje važi za dioide je fundamentalno (videti [78, 10]).

Propozicija 1.28. *Neka je $(S,+,\cdot,0,1)$ dioid. Tada važi sledeće:*

(i) *Kanoničko kvazi-uređenje, definisano sa* (1.59)*, kompatibilno je sa operacijama $+$ i $\cdot$, tj. za svako $x,y,z \in S$ važi sledeće:*

$$\begin{aligned} x \leq y \quad &\textit{povlači} \quad x+z \leq y+z, \\ x \leq y \quad &\textit{povlači} \quad x \cdot z \leq y \cdot z, \\ x \leq y \quad &\textit{povlači} \quad z \cdot x \leq z \cdot y, \end{aligned}$$

(ii) *0 je najmanji element u S, tj.*

$$0 \leq x, \quad \textit{za svako } x \in S.$$

Posebno bogatu klasu dioida čine aditivno idempotentni poluprsteni. Na aditivno idempotentnom poluprstenu S, kanoničko delimično uređenje može da se okarakteriše sa:

$$x \leq y \quad \text{akko} \quad x + y = y, \quad \text{za svako } x, y \in S, \tag{1.60}$$

Zaista, $x \leq y$ za $x, y \in S$ je, prema definiciji (1.59), ekvivalentno sa tim da postoji $z \in S$ tako da je $x + z = y$. Ili drugim rečima, $x + y = x + x + z = x + z = y$. Dakle, $x \leq y$ akko $x + y = y$. Štaviše, ako je S aditivno idempotentan poluprsten, tada je $(S, \leq)$ supremum-polumreža sa najmanjim elementom 0, pri čemu je $\leq$ kanoničko delimično uređenje na S (ili ekvivalentno, gde je $x \vee y = x + y$).

Neka je A neprazan skup i S poluprsten. Tada je $A \times A$-*matrica* nad S svako preslikavanje $\mu : A \times A \to S$. Slično, A-*vektor* nad S je svako preslikavanje $\mu : A \to S$. Skup svih $A \times A$-matrica nad S označava se sa $S^{A \times A}$, dok se skup svih A-vektora nad S označava sa S^A. Za matricu $\mu \in S^{A \times B}$, njena *transponovana matrica* $\mu^{\mathrm{T}} \in S^{B \times A}$ definiše se sa $\mu^{\mathrm{T}}(y, x) = \mu(x, y)$, za svako $x, y \in A$. *Nula matrica* $0_A \in S^{A \times A}$ definiše se sa $0_A(a, b) = 0$ za svako $a, b \in A$, *jedinična* (ili *identička*) matrica $I_A \in S^{A \times A}$ sa $I_A(a, a) = 1$ za svako $a \in A$ i $I_A(a, b) = 0$ za svako $a, b \in A$ za koje je $a \neq b$. Na kraju, *univerzalna matrica* $U_A \in S^{A \times A}$ definiše se sa $U_A(a, b) = 1$ za svako $a, b \in A$.

Neka je $\mu \in S^{A \times A}$ proizvoljna $A \times A$-matrica nad S i $a \in A$ element iz A. Tada se definiše *a-ti red-vektor* $a\mu \in S^A$ sa $a\mu(x) = \mu(a, x)$, kao i *a-ti kolona-vektor* $\mu b \in S^A$ sa $\mu a(x) = \mu(x, a)$, za svako $x \in A$.

Za dve $A \times A$-matrice $\mu_1, \mu_2 \in S^{A \times A}$, njihova *matrična suma* $\mu_1 + \mu_2 \in S^{A \times A}$ i *Hadamardov proizvod* $\mu_1 \odot \mu_2 \in S^{A \times A}$ definišu se pokoordinatno, tj. za svako $x, y \in A$ važi

$$(\mu_1 + \mu_2)(a, b) = \mu_1(a, b) + \mu_2(a, b),$$
$$(\mu_1 \odot \mu_2)(a, b) = \mu_1(a, b) \cdot \mu_2(a, b),$$

dok se njihov *matrični proizvod* $\mu_1 \cdot \mu_2 \in S^{A \times A}$ definiše za svako $a, c \in A$ sa

$$(\mu_1 \cdot \mu_2)(a, c) = \sum_{b \in A} \mu_1(a, b) \cdot \mu_2(b, c).$$

U slučaju kada je $\mu_1 \in S^A$ i $\mu_2 \in S^{A \times A}$, tada se definiše njihov *matrično-vektorski proizvod* $\mu_1 \cdot \mu_2 \in S^A$ za svako $a \in A$ sa:

$$(\mu_1 \cdot \mu_2)(a) = \sum_{b \in A} \mu_1(a, b) \cdot \mu_2(b).$$

Slično, kada je $\mu_1 \in S^{A \times A}$ i $\mu_2 \in S^A$, tada se njihov *matrično-vektorski proizvod* $\mu_1 \cdot \mu_2 \in S^A$ definiše za svako $b \in A$ sa:

$$(\mu_1 \cdot \mu_2)(b) = \sum_{a \in A} \mu_1(a) \cdot \mu_2(a, b).$$

Na kraju, kada je $\mu_1 \in S^A$ i $\mu_2 \in S^A$, njihov *skalarni proizvod* $\mu_1 \cdot \mu_2 \in S$ definiše se sa:

$$\mu_1 \cdot \mu_2 = \sum_{a \in A} \mu_1(a) \cdot \mu_2(a).$$

Takođe, za matricu $\mu \in S^{A \times A}$ i $n \in \mathbb{N}_0$, *n-ti matrični stepen* od μ definiše se kao matrica $\mu^n \in S^{A \times A}$ induktivno sa:

$$\mu^0 = I_A,$$

$$\mu^{n+1} = \mu^n \cdot \mu.$$

Propozicija 1.29. *Neka je A neprazan konačan skup, S poluprsten, $\alpha, \beta \in S^A$, $\mu, \nu, \eta \in S^{A\times A}$ i $\{\mu_i\}_{i\in I} \subseteq S^{A\times A}$. Tada važe sledeća svojstva:*

$$\mu + \nu = \nu + \mu, \tag{1.61}$$
$$(\mu + \nu) + \eta = \mu + (\nu + \eta), \tag{1.62}$$
$$(\mu \cdot \nu) \cdot \eta = \mu \cdot (\nu \cdot \eta), \tag{1.63}$$
$$\alpha \cdot (\mu \cdot \nu) = (\alpha \cdot \mu) \cdot \nu,\ \alpha \cdot (\mu \cdot \beta) = (\alpha \cdot \mu) \cdot \beta,\ (\mu \cdot \nu) \cdot \alpha = \mu \cdot (\nu \cdot \alpha), \tag{1.64}$$
$$\left(\sum_{i\in I} \mu_i\right) \cdot \nu = \sum_{i\in I} \mu_i \cdot \nu, \quad \nu \cdot \left(\sum_{i\in I} \mu_i\right) = \sum_{i\in I} \nu \cdot \mu_i \tag{1.65}$$
$$(\mu \cdot \nu)^{\mathrm{T}} = \nu^{\mathrm{T}} \cdot \mu^{\mathrm{T}}. \tag{1.66}$$

Stoga se sve zagrade u (1.62) – (1.64) mogu izostaviti. Takođe, zaključujemo da je struktura $(S^{A\times A}, +, \cdot, 0_A, I_A)$ poluprsten.

Neka je S aditivno idempotentan poluprsten. Tada se definiše *uređenje* dve $A \times A$-matrice $\mu_1, \mu_2 \in S^{A\times A}$ pokoordinatno, odnosno:

$$\mu_1 \leq \mu_2 \quad \text{akko} \quad \mu_1(a,b) \leq \mu_2(a,b), \tag{1.67}$$

za svako $a, b \in A$. Ovo uređenje kompatibilno je sa matričnom sumom, proizvodom i transpozicijom, tj. važe sledeća tvrđenja.

Propozicija 1.30. *Neka je A neprazan konačan skup i S aditivno idempotentni poluprsten. Tada za svako $\mu_1, \mu_2, \nu_1, \nu_2 \in S^{A\times A}$ važi:*

$$\mu_1 \leq \mu_2 \ \textit{i}\ \nu_1 \leq \nu_2 \quad \textit{povlači} \quad \mu_1 + \nu_1 \leq \mu_2 + \nu_2,$$
$$\mu_1 \leq \mu_2 \ \textit{i}\ \nu_1 \leq \nu_2 \quad \textit{povlači} \quad \mu_1 \cdot \nu_1 \leq \mu_2 \cdot \nu_2,$$
$$\mu_1 \leq \mu_2 \quad \textit{povlači} \quad \mu_1^{\mathrm{T}} \leq \mu_2^{\mathrm{T}}.$$

Za svaki aditivno idempotentan poluprsten S i neprazan skup A, uređena petorka $(S^{A\times A}, +, \cdot, 0_A, I_A)$ formira aditivno idempotentan poluprsten. Štaviše, prirodno uređenje na ovom aditivno idempotentnom poluprstenu, definisano pravilom (1.60), poklapa se sa pokoordinatnim uređenjem matrica (1.67).

Matrice nad Boolovim poluprstenom nazivaju se *Boolove matrice*. Skup svih $A \times A$-Boolovih matrica označava se sa $2^{A\times A}$. Za aditivno idempotentan poluprsten S, skup $\{0,1\}$ formira Boolov poluprsten od S. Stoga, matrice nad S koje uzimaju vrednosti u skupu $\{0,1\}$ mogu se identifikovati sa Boolovim matricama, tj. može se reći da je $2^{A\times A}$ podpoluprsten aditivno idempotentnog poluprstena $S^{A\times A}$.

Štaviše, postoji bijektivna korespodencija između skupa Boolovih matrica $2^{A\times A}$ i skupa svih binarnih relacija na skupu A. Zaista, za $\mu \in 2^{A\times A}$ i $R \subseteq A \times A$, važi $(x,y) \in R$ akko je $\mu(x,y) = 1$, kao i $(x,y) \notin R$ akko je $\mu(x,y) = 0$. Povrh toga, uređenje Boolovih matrica odgovara inkluziji binarnih relacija, matrična suma i Hadamardov proizvod Boolovih matrica odgovaraju uniji i preseku binarnih relacija, dok matrični proizvod Boolovih matrica odgovara proizvodu (kompoziciji) binarnih relacija. Stoga se ne pravi razlika između skupa svih $A \times A$-Boolovih matrica i skupa svih binarnih relacija na A, te oba skupa označavamo istom oznakom $2^{A\times A}$. Na kraju, za Boolove matrice $\mu, \nu \in 2^{A\times A}$ definiše se *razlika* kao Boolova matrica $\varrho = \mu - \nu \in 2^{A\times A}$ za koju važi $\varrho + \nu = \mu$. Očigledno, razlika između dve Boolove matrice odgovara razlici između dve binarne relacije.

Po uzoru na binarne relacije, za Boolovu matricu $\varrho \in 2^{A\times A}$ kaže se da je:

(1) *refleksivna*, ako je $\varrho(x,x) = 1$, za svako $x \in A$;

(2) *simetrična*, ako je $\varrho(x,y) = \varrho(y,x)$, za svako $x, y \in A$;

(3) *tranzitivna*, ako je $\varrho(x,y) \cdot \varrho(y,z) \leq \varrho(x,z)$, za svako $x, y, z \in A$.

Refleksivna i tranzitivna Boolova matrica naziva se *matrica kvazi-uređenja*, dok se simetrična matrica kvazi-uređenja naziva *matrica ekvivalencije*.

Lema 1.31. *Neka je $\varrho \in 2^{A\times A}$ matrice kvazi-uređenja. Tada je $\varrho \cdot \varrho = \varrho$.*

Lema 1.32. *Neka su $\varrho, \theta \in 2^{A\times A}$ matrice kvazi-uređenja (odnosno, matrice ekvivalencije). Tada je i $\varrho \odot \theta \in 2^{A\times A}$ takođe matrica kvazi-uređenja (odnosno, matrica ekvivalencija).*

Lema 1.33. *Neka su $\varrho, \theta \in 2^{A\times A}$ matrice kvazi-uređenja takve da je $\varrho \leq \theta$. Tada je $\varrho \cdot \theta = \theta \cdot \varrho = \theta$.*

Za matricu ekvivalencije $\varrho \in 2^{A\times A}$ i $a \in A$, a-ti red-vektor (ili a-ti kolona-vektor) naziva se *klasa ekvivalencije* od ϱ sa a, i piše se ϱ^a umesto $a\varrho$ (ili ϱa). Skup svih klasa ekvivalencija od ϱ označava se sa A/ϱ, i naziva se *faktor skup* od A u odnosu na ϱ. Kao i u slučaju binarnih relacija ekvivalencija na A, imamo da važi:

(1) $\varrho^a = \varrho^b$ kada je $\varrho(a,b) = 1$;

(2) $\varrho^a \odot \varrho^b = 0_A$ kada je $\varrho(a,b) = 0$, pri čemu je $0_A \in 2^A$ definisan sa $0_A(a) = 0$ za svako $a \in A$;

(3) $\sum_{a\in A} \varrho^a = I_A$, pri čemu je $I_A \in 2^A$ definisan sa $I_A(a) = 1$ za svako $a \in A$.

Neka su $\varrho, \theta \in 2^{A\times A}$ dve matrice kvazi-uređenja. Kaže se da je matrica ϱ *usitnjenje* matrice θ ako je $\varrho \leq \theta$. Drugim rečima, ϱ je usitnjenje θ ako je svaka red-vrsta od ϱ podskup neke red-vrste od θ (ili ekvivalentno, ako je svaka kolona-vrsta od ϱ podskup neke kolona-vrste od θ).

Neka je S aditivno idempotentan poluprsten, A neprazan skup, i neka su $\alpha, \beta \in S^{A\times A}$. Tada je *Boolov levi rezidual* od β sa α Boolova matrica $\beta/\alpha \in 2^{A\times A}$ definisana za svako $a, b \in A$ sa:

$$(\beta/\alpha)(b,a) = \lceil a\alpha \leq b\beta \rceil, \tag{1.68}$$

dok je *Boolov desni rezidual* od β sa α je Boolova matrica $\alpha\backslash\beta \in 2^{A\times A}$ definisana za svako $a, b \in A$ sa:

$$(\alpha\backslash\beta)(a,b) = \lceil \alpha a \leq \beta b \rceil, \tag{1.69}$$

Sledeći rezultat, vezan za Boolov levi i desni rezidual, dokazan je u [51].

Lema 1.34. *Neka su $\alpha, \beta \in 2^{A\times A}$. Tada za svako $\chi \in 2^{A\times A}$ važe sledeća svojstva adjunkcije:*

$$\chi \cdot \alpha \leq \beta \quad akko \quad \chi \leq \beta/\alpha, \tag{1.70}$$

$$\alpha \cdot \chi \leq \beta \quad akko \quad \chi \leq \alpha\backslash\beta. \tag{1.71}$$

Stoga je za svako $\alpha, \beta \in 2^{A\times A}$, prema (1.70), Boolov levi rezidual $\beta/\alpha \in 2^{A\times A}$ od β sa α najveće rešenje matrične nejednačine $\chi \cdot \alpha \leq \beta$ u skupu svih $A \times A$-Boolovih matrica, gde je $\chi \in 2^{A\times A}$ nepoznata Boolova matrica. Slično, prema (1.71), Boolov desni rezidual $\alpha\backslash\beta \in 2^{A\times A}$ od β sa α najveće rešenje matrične nejednačine $\alpha \cdot \chi \leq \beta$ u skupu svih $A \times A$-Boolovih matrica. Pored toga, definiše se i *Boolov rezidual* $\alpha|\beta \in 2^{A\times A}$ sa

$$\alpha|\beta = (\alpha\backslash\beta) \odot (\alpha/\beta).$$

Takođe, neka je $\nu, \eta \in S^A$. Tada su *Boolov levi rezidual* od η sa ν i *Boolov desni rezidual* od η sa ν su Boolove matrice $\eta/\nu \in 2^{A\times A}$ i $\nu\backslash\eta \in 2^{A\times A}$, respektivno, definisane sa

$$(\eta/\nu)(b,a) = (\nu\backslash\eta)(a,b) = \lceil \nu(a) \leq \eta(b) \rceil,$$

za svako $a, b \in A$, odnosno važi $(\eta/\nu) = (\nu\backslash\eta)^{\mathrm{T}}$. Slično kao u Lemi 1.34, može se pokazati da je Boolov levi rezidual $\eta/\nu \in 2^{A\times A}$ od η sa ν najveće rešenje matrične nejednačine $\chi \cdot \nu \leq \eta$ u skupu svih $A \times A$-Boolovih matrica, gde je $\chi \in 2^{A\times A}$ nepoznata Boolova matrica. Takođe, Boolov desni rezidual $\nu\backslash\eta \in 2^{A\times A}$ od η sa ν je najveće rešenje matrične nejednačine $\nu \cdot \chi \leq \eta$ u skupu svih $A \times A$-Boolovih matrica, gde je $\chi \in 2^{A\times A}$ nepoznata Boolova matrica. Pored toga, definiše se i *Boolov birezidual* $\nu|\eta \in 2^{A\times A}$ sa $\nu|\eta = (\nu\backslash\eta) \odot (\nu/\eta)$, ili ekvivalentno,

$$(\nu|\eta)(a,b) = \lceil \nu(a) = \eta(b) \rceil, \tag{1.72}$$

za svako $a, b \in A$. Sledeće osobine Boolovih reziduala koriste se u daljem radu.

Lema 1.35. *a) Neka su $\varrho, \theta \in 2^{A\times A}$ matrice kvazi-uređenja takve da je ϱ usitnjenje θ. Tada je ϱ takođe usitnjenje $\theta a/\theta a$, za svako $a \in A$.*

b) Neka su $\varrho, \theta \in 2^{A\times A}$ matrice ekvivalencije takve da je ϱ usitnjenje θ. Tada je ϱ usitnjenje $\theta^a|\theta^a$, za svako $a \in A$.

Dokaz. a) S obzirom da je $\varrho \leq \theta$ i θ matrica kvazi-uređenja, sledi da je $\varrho \cdot \theta \leq \theta \cdot \theta = \theta$. Neka su $a, b \in A$.Tada važi:

$$(\varrho \cdot \theta)(b,a) = \sum_{c\in A} \varrho(b,c) \cdot \theta(c,a) = \sum_{c\in A} \varrho(b,c) \cdot \theta a(c) = (\varrho \cdot \theta a)(b), \tag{1.73}$$

stoga imamo $\varrho \cdot \theta a \leq \theta a$, ili ekvivalentno, $\varrho \leq \theta a/\theta a$, za svako $a \in A$.

b) S obzirom da je ϱ i θ matrica ekvivalencije, iz a) imamo $\varrho \leq \theta^a/\theta^a$, za svako $a \in A$. Štaviše, iz simetričnosti ϱ sledi da za svako $a \in A$ važi $\varrho = \varrho^{\mathrm{T}} \leq (\theta^a/\theta^a)^{\mathrm{T}} = \theta^a\backslash\theta^a$. Dakle, $\varrho \leq \theta^a|\theta^a$, za svako $a \in A$. □

Glava 2

Fazi i težinski automati

Nedugo nakon uvođenja teorije fazi skupova u radu Zadeh [219] iz 1965. godine, teorija fazi automata nastaje kao odgovor na potrebu stvaranja matematičkog modela izračunljivosti koji je u stanju da uključi pojmove koji se često susreću u izučavanju prirodnih jezika, poput "neodređenosti" i "nepreciznosti". Prvu matematičku formulaciju fazi automata dao je Wee [209] 1967. godine, a kasnije i Wee i Fu [210]. Nedugo zatim, Santos [187] je 1968. godine definisao takozvane maxmin automate (takođe videti [188, 189, 186]), dok su Lee i Zadeh [131] definisali koncept fazi automata sa konačnim brojem stanja 1969. godine. Mizumoto i ostali [151, 152] potom su izučavali fazi automate sa fazi inicijalnim stanjem. Asai i Kitajima [7] izučavali su dve klase fazi automata koje odgovaraju Mealyjevom i Moorovom tipu običnih automata. Malik i ostali [139] uveli su novi tip fazi automata 1999. godine. Krithivasan i Sharda [125] opisali su pojam fazi ω-automata. Fazi automate nad Gödelovom strukturom intenzivno su izučavani do početka 2000-ih godina (videti [63, 83, 157] i reference u njima).

Prvi pokušaj definisanja fazi automata nad generalnim algebarskim strukturama dao je Wechler [208] 1978. godine. Od tada, fazi automati nad različitim uređenim stukturama izučavani su u velikom broju radova. Preciznije, fazi automate nad mrežno uređenim monoidima izučavali su Li i ostali [132, 133, 135, 137], dok su fazi automati nad mrežama izučavani u [14, 62, 124, 127, 134, 136, 163, 164, 165, 166]. Močkoř je u [153] predstavio fazi automate kao ugnježdene sisteme nedeterminističkih automata, dok je Bělohlávek u [14] predstavio determinističke automate sa fazi skupom završnih stanja kao ugnježdene sisteme determinističkih automata. Štaviše, Bělohlávek je razmatrao fazi automate nad lokalno konačnim kompletnim mrežama, i dokazao je da bilo koji fazi jezik koji je raspoznat fazi konačnim raspoznavačem takođe može biti raspoznat i determinističkim fazi konačnim raspoznavačem. Opštiji rezultat dat je u radu Li i Pedrycz [135]. U tom radu, autori su razmatrali fazi automate nad mrežno uređenim monoidom $\mathcal{L}$, i dokazali su da bilo koji fazi jezik, koji je raspoznat fazi konačnim raspoznavačem nad $\mathcal{L}$, može biti raspoznat determinističkim fazi konačnim raspoznavačem ako i samo ako je poluprstenski ostatak od $\mathcal{L}$ (u odnosu na operacije supremum i množenje) lokalno konačan. Pored toga, Jin i ostali su u [113] izučavali fazi automate nad po-monoidima, Li u [134] nad generalnim mrežama. Fazi automate nad kompletnim reziduiranim mrežama izučavali su Qui i saradnici u [173, 174, 213, 214, 215, 216, 217]. Sa druge tačke gledišta, fazi automate nad kompletnim reziduiranim mrežama izučavali su Ignjatović, Ćirić i saradnici [41, 42, 45, 97, 100, 108, 109, 110, 147, 194, 197].

Kao što je dobro poznato, u "krisp" slučaju postoji samo jedan model determinističkog automata, i zadaje se skupom stanja, jedinstvenim početnim stanjem, funkcijom prelaza (koje slika svako stanje i simbol u neko stanje) i skupom završnih stanja kao podskup skupa stanja. Takođe je poznato da i za svaki nedeterministički konačni automat postoji ekvivalentan deterministički konačan automat. U fazi slučaju,

postoji više modela determinističkih fazi automata. Verovatno najizučavaniji model determinističkog fazi automata je takozvani *krisp-deterministički fazi automat* (k-DFA, skraćeno), koji predstavlja deterministički automat u kojem je skup završnih stanja zamenjen fazi podskupom skupa stanja fazi automata. Takav model fazi automata najpre je izučan u radu Bělohlávek [14], a potom i Li i Pedrycz [135]. U istim radovima razvijeni su i determinizacioni metodi za konverziju fazi automata u ekvivalentan k-DFA. Drugi determinizacioni metod razvili su Ignjatović i ostali [97], da bi kasnije u [100] pokazali da k-DFA može alternativno biti konstruisan preko Nerodovih desnih kongurencija originalnog fazi automata. k-DFA-i nadalje su izučavani u mnogobrojnim radovima [109, 110, 147, 196, 202, 206].

Iako imaju vrlo prirodnu definiciju, glavni nedostatak k-DFA-a leži u tome što su u stanju da raspoznaju jedino fazi jezike sa konačnim rangom. Stoga, svi determinizacioni metodi razvijeni u prethodno nabrojanim radovima ne mogu se primeniti na fazi automate koji raspoznaju fazi jezike beskonačnog ranga. Stoga, de Mendívil i Garitagoitia uvode u [145] novi model determinističkog fazi automata, takozvani *kompletan deterministički fazi automat* (KDFA, skraćeno), koji predstavlja generalizaciju k-DFA. Preciznije, KDFA je fazi automat koji ima jedinstveno početno stanje sa određenim stepenom, koji ima prelaz iz svakog stanja i za svaki simbol u drugo stranje u određenom stepenu, kao i skup završnih stanja kao fazi podskup skupa stanja fazi automata. Ovakav deterministički model fazi automata u stanju je da raspozna fazi jezik sa beskonačnim rangom. U istom radu, autori su dali takozvanu *Lemu o napumpavanju* za fazi jezike raspoznate od KDFA, te time odredili neophodan uslov za determinizaciju fazi automata. U radu [144], isti autori dali su determinizacioni algoritam za konverziju fazi automata u ekvivalentan KDFA.

Ideja za definiciju i konstrukciju KDFA potekla je zapravo iz literature težinskih automata. Nakon Schützenbergera [191], koji je prvi dao njihovu osnovnu karakterizaciju, težinski automati dolaze u fokus teoretskih istraživanja već ranih sedamdesetih godina [18, 64, 184], a nedugo potom nalaze i primenu u brojnim poljima, između ostalih, u automatskom prepoznavanju glasa, kompresiji slika, diskretnim sistemima događajima, lingvistici, procesuiranju prirodnih jezika (videti [10, 31, 71, 72, 85, 87, 102, 112, 115, 154, 155, 168] i reference u njima). Težinski automati izučavani su nad brojnim strukturama, preciznije, nad tropskim poluprstenima [154], min-plus i max-plus algebrama [13, 122, 10], poluprstenima [116, 126, 60], hemiprstenima [59], jakim bimonoidima [61, 43, 109]. Model determinističkog automata dali su Kirsten i Mäurer u [116] za težinske automate nad poluprstenima. Njihov automat, kao i detertminizacioni algoritam, predstavljaju generalizaciju rezultata vezanih za težinske automate nad tropskim poluprstenom (videti [2, 154] i [60, Poglavlje 7.2]). Iako težinski automati predstavljaju opštiji pojam od fazi automata, jer su strukture nad kojima se obično definišu težinski automati opštije od struktura nad kojima se obično definišu fazi automati jer ne zahtevaju uređenje (videti [60]), pokazaćemo da se mnogi rezultati vezani za determinizaciju fazi automata mogu primeniti uz određene adaptacije i na težinske automate nad određenim tipovima dioida.

Glava je organizovana na sledeći način. U Sekciji 2.1 date su definicije fazi automata i fazi jezika nad kompletnim reziduiranim mrežama, kao i težinskih automata i formalnih stepenih redova nad poluprstenima. Sekcija 2.2 posvećena je grafičkoj reprezentaciji i analizi fazi i težinskih automata. Na kraju, u Sekciji 2.3 date su definicije kompletnih determinističkih fazi i težinskih automata, kao i krisp-determinističkih fazi automata kao poseban slučaj KDFA-a.

2.1 Fazi i težinski automati

Kao i ranije, neprazan skup simbola koji nazivamo alfabet označava se sa X, dok se slobodna polugrupa i slobodan monoid nad alfabetom X označavaju sa X^+ i X^*, redom. Prazna reč označava se sa ε. Na kraju, $\mathcal{L}$ označava kompletnu reziduiranu mrežu.

Fazi automat nad $\mathcal{L}$ i X, ili jednostavno samo *fazi automat*, je uređena četvorka $\mathcal{A} = (A, \sigma, \delta, \tau)$, pri čemu je:

(1) A neprazan skup, koji se naziva *skup stanja*,

(2) $\sigma : A \to L$ fazi podskup od A, koji se naziva *fazi skup početnih stanja*,

(3) $\delta : A \times X \times A \to L$ fazi podskup od $A \times X \times A$, koji se naziva *funkcija fazi prelaza*,

(4) $\tau : A \to L$ fazi podskup od A, koji se naziva *fazi skup završnih stanja*.

Za fazi automat $\mathcal{A}$, $\sigma(a)$ interpretira se kao stepen u kojem je $a \in A$ početno stanje, $\delta(a, x, b)$ kao stepen u kojem ulazni simbol $x \in X$ prouzrokuje prelaz iz stanja $a \in A$ u stanje $b \in A$, a $\tau(a)$ kao stepen u kojem je $a \in A$ završno stanje. Takođe, za stanje $a \in A$ kaže se da je *početno* (odnosno, *završno*) ako je $\sigma(a) > 0$ (odnosno, $\tau(a) > 0$).

Zbog metodoloških razloga, dozvoljavamo da skup stanja A bude beskonačan. Fazi automat kod kojeg je skup stanja konačan naziva se *fazi konačni automat (FKA)*.

Ukoliko su σ i τ krisp podskupovi od A i δ krisp podskup od $A \times X \times A$, tada je $\mathcal{A}$ običan (krisp) nedeterministički automat, a ukoliko su σ i τ krisp podskupovi od A, i δ krisp preslikavanje $A \times X \to A$, tada je $\mathcal{A}$ običan (krisp) deterministički automat.

Funkcija fazi prelaza $\delta : A \times X \times A \to L$ može se proširiti na funkciju $\delta_* : A \times X^* \times A \to L$ na sledeći način: za praznu reč $\varepsilon \in X^*$ postavimo $\delta_*(a, \varepsilon, a) = 1$ za svako $a \in A$, i $\delta_*(a, \varepsilon, b) = 0$ za svako $a, b \in A$ pri čemu je $a \neq b$, kao i za svako $a, b \in A$, $u \in X^*$ i $x \in X$ postavimo da je:

$$\delta_*(a, ux, b) = \bigvee_{c \in A} \delta_*(a, u, c) \otimes \delta(c, x, b).$$

Funkcija fazi prelaza $\delta : A \times X \times A \to L$ takođe indukuje i familiju $\{\delta_u\}_{u \in X^*}$ fazi relacija na A sa:

$$\begin{aligned} &\delta_\varepsilon = \triangle_A, \\ &\delta_x(a, b) = \delta(a, x, b), \quad \text{za svako } a, b \in A \text{ i } x \in X, \\ &\delta_{ux} = \delta_u \circ \delta_x, \quad \text{za svako } u \in X^* \text{ i } x \in X. \end{aligned}$$

Indukcijom se dokazuje da za bilo koje dve reči $u, v \in X^*$ važi:

$$\delta_{uv} = \delta_u \circ \delta_v. \tag{2.1}$$

Takođe, ukoliko je $u = x_1 x_2 \dots x_n$, tada se može zapisati:

$$\delta_u = \delta_{x_1} \circ \delta_{x_2} \circ \dots \circ \delta_{x_n}. \tag{2.2}$$

Familija $\{\delta_u\}_{u \in X^*}$ naziva se *familija fazi relacija prelaza* od $\mathcal{A}$. Za svako $u = x_1 x_2 \dots x_n$, iz (2.2) sledi:

$$\delta_u(a, b) = \bigvee_{(c_1, \dots, c_{n-1}) \in A^{n-1}} \delta_{x_1}(a, c_1) \otimes \delta_{x_2}(c_1, c_2) \otimes \dots \otimes \delta_{x_n}(c_{n-1}, b) \tag{2.3}$$

$$= \bigvee_{(c_1,\ldots,c_{n-1})\in A^{n-1}} \delta(a, x_1, c_1) \otimes \delta(c_1, x_2, c_2) \otimes \ldots \otimes \delta(c_{n-1}, x_{n-1}, b). \quad (2.4)$$

Intuitivno, proizvod pod supremumom u (2.4) predstavlja stepen u kojem ulazna reč $u \in X^*$ prouzrokuje prelaz iz stanja $a \in A$ u stanje $b \in A$ kroz niz posrednih stanja $c_1, c_2, \ldots, c_{n-1} \in A$. Tada $\delta_u(a, b)$, kao supremum svih takvih stepena, predstavlja stepen u kojem ulazna reč u prouzrokuje prelaz iz stanja a u stanje b.

Na sličan način kao za familiju fazi relacija prelaza, definišu se dve familije fazi podskupova od A, $\{\sigma_u\}_{u\in X^*}$ i $\{\tau_u\}_{u\in X^*}$. Preciznije, za svako $u \in X^*$, definišu se fazi podskupovi σ_u i τ_u od A sa:

$$\sigma_u = \sigma \circ \delta_u, \quad (2.5)$$

$$\tau_u = \delta_u \circ \tau, \quad (2.6)$$

Iz (1.41), (1.42), (2.5) i (2.6) sledi da za svako $a \in A$ i $u \in X^*$ važi:

$$\sigma_u(a) = \bigvee_{b\in A} \sigma(b) \otimes \delta_u(b, a), \quad (2.7)$$

$$\tau_u(a) = \bigvee_{b\in A} \delta_u(a, b) \otimes \tau(b). \quad (2.8)$$

Proizvod na desnoj strani u (2.7) može se interpretirati kao stepen u kojem ulazna reč $u \in X^*$ prouzrokuje prelaz iz nekog početnog stanja b u stanje a, dok $\sigma_u(a)$, kao supremum svih takvih stepena, označava stepen postojanje takve putanje pod uticajem reči u. Stoga, σ_u predstavlja fazi skup takav da $\sigma_u(a)$ predstavlja stepen u kojem ulazna reč u prouzrokuje prelaz iz nekog početnog stanja fazi automata u stanje a. Na sličan način, na osnovu (2.8) zaključujemo da je τ_u fazi skup takav da $\tau_u(a)$ predstavlja stepen u kojem ulazna reč u prouzrokuje prelaz iz stanja a u neko završno stanje fazi automata. Stoga se familija $\{\sigma_u\}_{u\in X^*}$ naziva *familija fazi dostižnih stanja*, dok se familija $\{\tau_u\}_{u\in X^*}$ naziva *familija fazi ko-dostižnih stanja*.

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat. *Reverzni fazi automat od* $\mathcal{A}$ predstavlja fazi automat $\overline{\mathcal{A}} = (A, \bar{\sigma}, \bar{\delta}, \bar{\tau})$, gde je $\bar{\sigma} = \tau$, $\bar{\tau} = \sigma$, a $\bar{\delta} : A \times X \times A \to L$ definisano sa $\bar{\delta}(a, x, b) = \delta(b, x, a)$, za svako $a, b \in A$ i $x \in X$. Grubo govoreći, reverzni fazi automat $\overline{\mathcal{A}}$ od $\mathcal{A}$ dobijen je zamenom fazi skupova početnih i završnih stanja, kao i "zamenom" svih prelaza od $\mathcal{A}$. S obzirom da je operacija $\otimes$ komutativna, imamo da je:

$$\bar{\delta}_u(a, b) = \delta_{\bar{u}}(b, a)$$

za svako $a, b \in A$ i $u \in X^*$.

Fazi jezik nad $\mathcal{L}$ *i* X, ili jednostavno *fazi jezik*, je svako preslikavanje $f : X^* \to L$. Generalno, fazi jezik može imati beskonačan rang. *Fazi jezik raspoznat fazi automatom* $\mathcal{A} = (A, \sigma, \delta, \tau)$ je fazi jezik $[\![\mathcal{A}]\!] \in L^{X^*}$ definisan za svako $u \in X^*$ sa:

$$[\![\mathcal{A}]\!](u) = \bigvee_{a,b\in A} \sigma(a) \otimes \delta_u(a, b) \otimes \tau(b) = \sigma \circ \delta_u \circ \tau. \quad (2.9)$$

Drugim rečima, stepen pripadnosti reči u fazi jeziku $[\![\mathcal{A}]\!]$ jednak je stepenu u kojem fazi automat $\mathcal{A}$ prihvata ili raspoznaje reč u. Fazi automati $\mathcal{A}$ i $\mathcal{B}$ su *jezički ekvivalentni*, ili samo *ekvivalentni*, ako je $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.

Reverzni fazi jezik fazi jezika $f \in X^*$ je fazi jezik $\overline{f} \in X^*$ definisan sa $\overline{f}(u) = f(\bar{u})$, za svako $u \in X^*$. S obzirom da je $\overline{(\bar{u})} = u$ za svako $u \in X^*$, imamo da je $\overline{(\overline{f})} = f$ za

svaki fazi jezik $f \in X^*$. Jednostavno je proveriti da za svaki fazi automat $\mathcal{A}$ važi:

$$[\![\overline{\mathcal{A}}]\!] = \overline{[\![\mathcal{A}]\!]},$$

odnosno, fazi jezik raspozat reverznim fazi automatom $\overline{\mathcal{A}}$ jednak je reverznom fazi jeziku $\overline{[\![\mathcal{A}]\!]}$ fazi jezika $[\![\mathcal{A}]\!]$ raspoznat fazi automatom $\mathcal{A}$.

Neka je S proizvoljan poluprsten. *Težinski konačni automat nad X i S*, ili jednotavno samo *težinski konačni automat (TKA)*, predstavlja uređenu četvorku $\mathcal{A} = (A, \delta, \sigma, \tau)$, pri čemu je:

- A konačan neprazan *skup stanja*,
- $\delta : A \times X \times A \to S$ *težinska funkcija prelaza*,
- $\sigma \in S^A$ *početni težinski vektor*,
- $\tau \in S^A$ *završni težinski vektor*.

Skup stanja A, iz metodoloških razloga, takođe može biti beskonačan, i tada se $\mathcal{A}$ naziva *težinski automat*. Za svako $x \in X$ definiše se *težinska matrica prelaza* $\delta_x \in S^{A\times A}$ sa:

$$\delta_x(a, b) = \delta(a, x, b), \quad \text{za svako } a, b \in A.$$

S obzirom da se težinski automati uvode na isti način kao i fazi automati, $A \times A$-matrica δ_u, kao i A-vektori σ_u i τ_u, za svako $u \in X^*$ definišu se na isti način kao i u slučaju fazi automata, s tim što je operacije $\vee$ i $\otimes$ zamenjuju operacijama $+$ i $\cdot$ iz poluprstena, tim redom.

Formalni stepeni red nad X i S, ili jednostavno samo *red*, je preslikavanje $f : X^* \to S$. *Ponašanje* težinskog automata $\mathcal{A} = (A, \delta, \sigma, \tau)$ predstavlja red $[\![\mathcal{A}]\!]$ definisan sa:

$$[\![\mathcal{A}]\!](u) = \sigma \cdot \delta_u \cdot \tau = \sum_{a,b\in A} \sigma(a) \cdot \delta_u(a, b) \cdot \tau(b), \tag{2.10}$$

za svako $u \in X^*$. Težinski automati $\mathcal{A}$ i $\mathcal{B}$ su ekvivalentni ukoliko imaju isto ponašanje, tj. ako je $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.

2.2 Grafička reprezentacija fazi i težinskih automata

Podsetimo se da se *usmereni graf* (ili jednostavno *graf* u ostatku izlaganja) definiše kao par $G = (V, E)$, pri čemu je V skup čvorova, a $E \subseteq V \times V$ skup grana, pri čemu za svako $e = (a, b) \in E$ znači da postoji usmerena grana iz čvora $a \in V$ u čvor $b \in V$. Dozvoljavamo da grane budu označene, odnosno, formalno govoreći, ukoliko postoje skupovi oznaka $X_1, X_2, \ldots, X_n$, tada se skup grana E definiše sa $E \subseteq V \times X_1 \times \ldots \times X_n \times V$, pri čemu za svako $e = (a, x_1, \ldots, x_n, b) \in E$ znači da postoji usmerena grana iz čvora $a \in V$ u čvor $b \in V$ označena oznakama $x_1 \in X_1, \ldots, x_n \in X_n$.

Za dati graf $G = (V, E)$, *putanja* u grafu G definiše se kao niz stanja tako da postoji grana između svaka dva stanja u nizu. *Ukorenjeno stablo* (ili samo *stablo* u nastavku izlaganja) je graf u kojem postoji tačno jedna putanja između bilo koja dva čvora grafa, i postoji poseban čvor u grafu koji sa naziva *koren stabla*. Tada su sve grane u stablu orijentisane iz korena stabla ka ostalim čvorovima. Za dato $n \in \mathbb{N}$, *n-arno stablo* predstalja stablo u kojem svaki čvor ima najviše n potomaka, dok je *puno n-arno stablo* stablo u kojem svaki čvor ima tačno n potomaka.

FKA $\mathcal{A} = (A, \sigma, \delta, \tau)$ može se vizuelno predstaviti kao označen usmeren graf (A, E), ciji su čvorovi stanja iz skupa A, i $E \subseteq A \times X \times L \times A$, pri čemu postoji

usmerena grana $e = (a, x, \delta_x(a,b), b) \in E$ iz čvora a u čvor b označena sa $x/\delta_x(a,b)$, ako postoji $x \in X$ tako da je $\delta_x(a,b) > 0$. Za svako početno stanje a fazi automata $\mathcal{A}$ crta se strelica označena sa $\sigma(a)$ koja ulazi u čvor a, dok za svako završno stanje a fazi automata $\mathcal{A}$ odgovarajući čvor a u grafu predstavlja se dvostrukim krugom i označava se sa $\tau(a)$. Takođe, obično se stepeni $\delta_x(a,b)$, $\sigma(a)$ i $\tau(a)$ ne pišu ukoliko su jednaki 1. Ovakav graf naziva se *graf prelaza* fazi automata $\mathcal{A}$.

U grafu prelaza fazi automata $\mathcal{A}$ definiše se *putanja preko reči* u (ili jednostavno *putanja* ako je reč u jasna iz kontektsta), gde je $u = x_1 x_2 \dots x_n$ i $n \geq 0$, kao torka $\pi_u = (a_1, \dots, a_{n+1}) \in A^{n+1}$ takva da je $\delta_{x_i}(a_i, a_{i+1}) > 0$, za svako $1 \leq i \leq n$. Izraz

$$w(\pi_u) = \bigotimes_{i=1}^{n} \delta_{x_i}(a_i, a_{i+1})$$

naziva se *težina* putanje π_u. Pri tome je uvek $w(\pi_u) > 0$ ako je $\mathcal{L}$ bez delioca nule. *Početno stanje putanje* π_u je stanje a_1, i označava se sa $s\pi_u$, dok je *završno stanje putanje* π_u stanje a_{n+1}, i označava se sa $e\pi_u$. *Dužina* putanje π_u poklapa se sa dužinom reči u. U konkretnom slučaju kada je $u = \varepsilon$, postoji putanja $\pi_\varepsilon = (a, a)$ za svako $a \in A$, s obzirom da je, prema definiciji, $\delta_\epsilon(a,a) = 1$ za svako $a \in A$. Putanja može biti i beskonačna, i pritom, smatramo da putanja nema završno stanje.

Generalno, u grafu prelaza fazi automata $\mathcal{A}$ može postojati više putanja preko reči u, te se skup svih putanja preko reči u označava sa $P_u^{\mathcal{A}}$ (ili jednostavno sa P_u kada je fazi automat $\mathcal{A}$ jasan iz konteksta). Za dve putanje $\pi_u = (a_1, a_2, \dots, a_{n+1}) \in P_u$ i $\pi_v = (b_1, b_2, \dots, b_{m+1}) \in P_v$, njihovo *nadovezivanje* ili *konkatenacija* je moguća ako je $e\pi_u = s\pi_v$, a rezultat nadovezivanja je putanja $\pi_{uv} = (a_1, a_2, \dots, a_{n+1}, b_1, b_2, \dots, b_{m+1}) \in P_{uv}$ čija je težina $w(\pi_{uv}) = w(\pi_u) \otimes w(\pi_v)$.

Za svaku reč $u \in X^*$ definiše se njen *stepen prihvatanja* od fazi automata $\mathcal{A}$ preko sledećeg izraza:

$$\bigvee_{\pi_u \in P_u} \sigma(s\pi_u) \otimes w(\pi_u) \otimes \tau(e\pi_u). \tag{2.11}$$

Izraz (2.11) može biti jednak 0 kada ne postoji putanja preko reči u (odnosno, kada je $P_u = \emptyset$), ili kada ne postoji putanja π_u u nepraznom skupu P_u takva da povezuje neko početno stanje i završno stanje fazi automata $\mathcal{A}$ (ili drugim rečima, kada je $\sigma(s\pi_u) = 0$ ili $\tau(e\pi_u) = 0$).

Sledeća Lema dokazana je u radu de Mendívil i Garitagoitia [145], a u njoj je iskazano da je stepen prihvatanja reči u od fazi automata $\mathcal{A}$ jednak stepenu pripadnosti reči u fazi jeziku $[\![\mathcal{A}]\!]$.

Lema 2.1. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat. Tada za svako $u \in X^*$ važi:*

$$[\![\mathcal{A}]\!](u) = \bigvee_{\pi_u \in P_u} \sigma(s\pi_u) \otimes w(\pi_u) \otimes \tau(e\pi_u).$$

Takođe, u grafu prelaza fazi automata $\mathcal{A}$ definiše se *ciklus preko reči* u (ili samo *ciklus* kada je reč u jasna iz konteksta) kao putanja preko reči u u kojoj se početno stanje i završno stanje putanje poklapaju. Ciklus preko reči u označava se sa ϱ_u. Napomenimo da ciklus preko reči u postoji ako je $|u| > 0$. Sa $\varrho_u^k = \varrho_u \dots \varrho_u$ (k puta) označava se putanja preko reči $u^k = u \cdot \dots \cdot u$ (k puta) koja se dobija nadovezivanjem ciklusa ϱ_u k puta (u smislu nadovezivanja putanja). Prema dogovoru, $\varrho_u^0 = \pi_\varepsilon$. Jasno je da je težina putanje ϱ_u^k jednaka

$$w(\varrho_u^k) = w(\varrho_u)^k,$$

za svako $k \geq 0$. Takođe važi i sledeće svojstvo (videti [145]).

Lema 2.2. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat i $u, v, w \in X^*$ tako da je $|v| > 0$. Ako je $\pi_{uvw} = \pi_u \varrho_v \pi_w$ (odnosno, π_{uvw} je putanja preko reči uvw dobijena nadovezivanjem putanje π_u preko u, ciklusa ϱ_v preko v, i putanje π_w preko w), tada je*

$$w(\pi_{uv^k w}) = w(\pi_u \varrho_v^k \pi_w) = w(\pi_u) \otimes w(\varrho_v)^k \otimes w(\pi_w),$$

za svako $k \geq 0$.

Analogno, svi pojmovi i oznake važe i za težinske automate, s tim što se operacije $\vee$ i $\otimes$ zamenjuju operacijama $+$ i $\cdot$ iz poluprstena, tim redom.

2.3 Deterministički fazi i težinski automati

Fazi (resp. težinski) automat je nedeterministički po svojoj prirodi. Zaista, za dato stanje u fazi (resp. težinskom) automatu i ulazni simbol može postojati više prelaza ka različitim stanjima iz datog stanja, svaki sa stepenom prelaza različitim od nule. U onome što sledi, prezentuju se deterministički oblici fazi i težinskih automata.

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat. Tada se za $\mathcal{A}$ kaže da je *kompletan* ako zadovoljava sledeći uslov:

(C) Za svako $x \in X$ i $a \in A$ postoji $b \in A$ tako da je $\delta_x(a, b) \neq 0$,

dok se za $\mathcal{A}$ kaže da je *deterministički* ako zadovoljava sledeća dva uslova:

(D1) Postoji jedinstveno stanje $a \in A$ tako da je $\sigma(a) \neq 0$,
(D2) Za svako $x \in X$ i $a, b_1, b_2 \in A$, ako je $\delta_x(a, b_1) \neq 0$ i $\delta_x(a, b_2) \neq 0$, tada je $b_1 = b_2$.

Ako je fazi automat $\mathcal{A}$ istovremeno i kompletan i deterministički, označava se sa $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$, tj. naglašava se njegovo jedinstveno početno stanje sa stepenom, i naziva se *kompletan deterministički fazi automat (KDFA)*. Kada je $\mathcal{A}$ konačan, tada se naziva *kompletan deterministički fazi konačni automat (KDFKA)*. Za fazi jezik $f : X^* \to L$ kaže se da je *KDFA-raspoznatljiv* ako postoji KDFA $\mathcal{A}$ takav da je $f = [\![\mathcal{A}]\!]$. U tom slučaju takođe se kaže i da KDFA $\mathcal{A}$ *raspoznaje* f.

Ako je $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ KDFA, tada za svaku reč $u = x_1 x_2 \dots x_n \in X^*$ postoji jedinstven niz stanja $a_\varepsilon, a_{x_1}, \dots, a_{x_1 x_2 \dots x_n}$ sa $a_\varepsilon = a$ tako da je $\delta_{x_i}(a_{x_1 \dots x_{i-1}}, a_{x_1 \dots x_i}) > 0$ za svako $1 \leq i \leq n$. Uopše, za svako $u = x_1 x_2 \dots x_n \in X^*$ i $b \in A$, označimo sa b_u stanje iz A koje je dostižno iz stanja b preko reči u preko jedinstvenog niza stanja $b_\varepsilon, b_{x_1}, \dots, b_{x_1 \dots x_n}$. Kao i obično, $b_\varepsilon = b$, za svako $b \in A$, tj. svako stanje je dostižno preko prazne reči ε. Tada (2.9) postaje:

$$[\![\mathcal{A}]\!](u) = \sigma(a) \otimes \left(\bigotimes_{i=1}^{n} \delta(a_{x_1 \dots x_{i-1}}, x_i, a_{x_1 \dots x_i}) \right) \otimes \tau(a_u). \tag{2.12}$$

Za KDFA $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ preslikavanje $\delta : A \times X \times A \to L$ može se proširiti na preslikavanje $\delta_* : A \times X^* \times A \to L$ na sledeći način: $\delta_*(a, \varepsilon, a) = 1$ za svako $a \in A$, i $\delta_*(a, \varepsilon, b) = 0$ za svako $a, b \in A$ za koje je $a \neq b$, kao i za svako $a, b \in A$, $u \in X^*$ i $x \in X$:

$$\delta_*(a, ux, b) = \begin{cases} \delta_*(a, u, c) \otimes \delta(c, x, b), & c = a_u \text{ i } b = a_{ux}, \\ 0, & \text{inače.} \end{cases} \tag{2.13}$$

Drugim rečima, za svako $u = x_1x_2 \dots x_n \in X^*$ gde je $n \geq 0$ i $a, b \in A$ imamo da je:

$$\delta_*(a, u, b) = \begin{cases} \bigotimes_{i=1}^{n} \delta(a_{x_1 \dots x_{i-1}}, x_i, a_{x_1 \dots x_i}), & \text{ako je } b = a_{x_1 \dots x_n}, \\ 0, & \text{inače.} \end{cases} \tag{2.14}$$

Primetimo da, ukoliko je fazi automat $\mathcal{A}$ kompletan a $\mathcal{L}$ bez delioca nule, tada je $\delta_*(a, u, a_u) > 0$ za svako $a \in A$ i $u \in X^*$. Štaviše, prema asocijativnosti $\otimes$ imamo da je $\delta_*(a, uv, a_{uv}) = \delta_*(a, u, a_u) \otimes \delta_*(a_u, v, a_{uv})$ za svako $a \in A$ i $u, v \in X^*$. Takođe, (2.12) može se zapisati i kao:

$$[\![\mathcal{A}]\!](u) = \sigma(a) \otimes \delta_*(a, u, a_u) \otimes \tau(a_u), \tag{2.15}$$

za svako $u \in X^*$.

Za KDFA $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$, ne postoji supremum u (2.7) i (2.8), te su familija fazi dostižnih stanja $\{\sigma_u\}_{u \in X^*}$ i familija fazi ko-dostižnih stanja $\{\tau_u\}_{u \in X^*}$ za svaki KDFA definisane za svako $u, v \in X^*$ preko:

$$\sigma_u(a_v) = \sigma(a_v) \otimes \delta_*(a_v, u, a_{vu}), \quad \tau_u(a_v) = \delta_*(a_v, u, a_{vu}) \otimes \tau(a_{vu}). \tag{2.16}$$

Neka je $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ KDFA. Za stanje $b \in A$ kaže se da je *dostižno* ako postoji $u \in X^*$ tako da je $\delta_*^A(a, u, b) > 0$. Ukoliko je stanje $b \in A$ dostižno, tada je iz (2.14), $b = a_u$. Drugim rečima, stanje b je dostižno iz početnog stanja a preko reči u. Ukoliko je svako stanje fazi automata $\mathcal{A}$ dostižno, tada se $\mathcal{A}$ naziva *dostižni KDFA*.

Ako fazi automat $\mathcal{A}$ zadovoljava sledeća dva uslova:

(C′) Za svako $x \in X$ i $a \in A$ postoji $a' \in A - \{a\}$ tako da je $\delta_x(a, a') = 1$ i $\delta_x(a, b) = 0$, za svako $b \in A - \{a, a'\}$,

(D′) Postoji $a_0 \in A$ tako da je $\sigma(a_0) = 1$ i $\sigma(a) = 0$ za svako $a \in A - \{a_0\}$,

tada se $\mathcal{A}$ naziva *krsip-deterministički fazi automat (k-DFA)*, a ako je dodatno skup A konačan, tada se $\mathcal{A}$ naziva *krsip-deterministički fazi konačni automat (k-DFKA)*. k-DFA može se zadati kao torka $\mathcal{A} = (A, a_0, \delta, \tau)$, gde je $\delta : A \times X \to A$ funkcija prelaza, $a_0 \in A$ jedinstveno početno stanje, i $\tau \in L^A$ fazi skup završnih stanja. Primetimo da se definicija k-DFA razlikuje od definicije determinističkog automata samo kod skupa završnih stanja - kod običnih determinističkih automata, skup završnih stanja je krisp podskup, dok je kod k-DFA skup završnih stanja fazi poskup od A. Ova činjenica dozvoljava da k-DFA prihvata reči u određenom stepenu, odnosno da raspoznaje fazi jezik.

Za fazi jezik $f : X^* \to L$ kaže se da je *k-DFA-raspoznatljiv* ako postoji k-DFA $\mathcal{A} = (A, a_0, \delta, \tau)$ takav da je $f = [\![\mathcal{A}]\!]$. U tom slučaju takođe se kaže i da k-DFA $\mathcal{A}$ *raspoznaje* f. Fazi jezik koji raspoznaje k-DFA $\mathcal{A}$ zadat je izrazom $[\![\mathcal{A}]\!](u) = \tau(\delta_*(a_0, u))$, za svako $u \in X^*$. Preciznije, rang fazi jezika $[\![\mathcal{A}]\!]$, gde je $\mathcal{A}$ k-DFA, sadržan je u rangu fazi podskupa τ, koji je konačan kada je skup stanja A konačan. Dakle, k-DFKA raspoznaje jedino fazi jezike sa konačnim rangom. Sa druge strane, KDFKA je, preko formule (2.12), u stanju da raspozna fazi jezik sa beskonačnim rangom.

Nerodov fazi automat od fazi automata $\mathcal{A} = (A, \sigma, \delta, \tau)$ je k-DFA $\mathcal{A}^N = (A^N, \sigma_\varepsilon, \delta^N, \tau^N)$, pri čemu je $A^N = \{\sigma_u | u \in X^*\}$, a $\delta^N : A^N \times X \to A^N$ i $\tau^N : A^N \to L$ su definisane sa:

$$\delta^N(\sigma_u, x) = \sigma_{ux}, \quad \tau^N(\sigma_u) = \sigma_u \circ \tau,$$

za svako $u \in X^*$ i $x \in X$. Za više informacija o Nerodovom fazi automatu videti [97, 100].

Kompletan deterministički (ili u nekim izvorima, *sekvencijalni*) težinski automat (KDTA, skraćeno) predstavlja težinski automat koji zadovoljava uslove (C), (D1) i (D2) sa strane 41. KDTA označavamo sa $\mathcal{A} = (A, \delta, \{a_0/\sigma(a_0)\}, \tau)$, kao i u slučaju fazi automata. Ukoliko je $\mathcal{A}$ konačan, tada se naziva *kompletan deterministički težinski konačni automat (KDTKA)*. Ponašanje KDTA-a zadato je redom:

$$[\![\mathcal{A}]\!](u) = \sigma(a_0) \cdot \left(\prod_{i=1}^{n} \delta_{x_i}(a_{i-1}, a_i) \right) \cdot \tau(a_n), \tag{2.17}$$

za svako $u = x_1 x_2 \dots x_n \in X^*$, pri čemu je $a_1, a_2 \dots, a_n \in A$ jedinstven niz stanja tako da je $\delta_{x_i}(a_{i-1}, a_i) > 0$ za svako $i \in \{1, 2, \dots, n\}$.

Glava 3

Računanje najvećih desno i levo invarijantnih fazi kvazi-uređenja i fazi ekvivalencija

Ekvivalentnost ponašanja različitih sistema, zajedno sa algoritmima za proveru i računanje takve ekvivalentnosti, izučavana je u mnogim oblastima matematike i računarskih nauka pod različitim imenima, a najčešće kao *simulacije* i *bisimulacije*. Mnogi autori u brojnim postavkama izučavali su simulacije i bisimulacije, na primer, Park [161] i Milner [148, 150, 149] u teroriji paralelnog izračunavanja, van Benthem [17] u modalnoj logici, Forti i Honsell u teoriji skupova [68], kao i u mnogim drugim oblastima (videti [180, 185]). Posebno značajno mesto simulacije i bisimulacije zauzimaju u teoriji automata, ili preciznije, u teoriji nedeterminističkih [46, 57, 138], težinskih [30, 51], fazi [41, 42, 193, 211, 212] i probabilističkih automata [12]. Posmatrana između dva identična NKA-a, bisimulacije su izučavane pod imenom *desno i levo invarijantne relacije* [33, 34, 35, 36, 103, 104, 105, 106], koje su prvenstveno kvazi-uređenja ili ekvivalencije, a koriste se za modelovanje ekvivalentnosti stanja NKA, u smislu da dva stanja NKA pripadaju desno ili levo invarijantnom kvazi-uređenju ili ekvivalenciji ako ih je nemoguće razlikovati.

Desno i levo invarijantne fazi relacije nad fazi automatima takođe su dobro izučavane [44, 45, 197], a njihove definicije baziraju se na odgovarajućim definicijama krisp desno i levo invarijantnih relacija. Preciznije, desno i levo invarijantne fazi relacije definisane su kao rešenja određenih sistema fazi relacijskih jednačina i nejednačina (videti i [96, 98, 99, 101]), a koriste se za modelovanje ekvivalentnosti stanja FKA. Desno i levo invarijantna fazi kvazi-uređenja i fazi ekvivalencije primenjene su u determinizaciji i redukciji stanja FKA (videti [44, 45, 110, 147, 197, 198, 195] i reference u njima), i u uskoj su vezi sa simulacijama i bisimulacijama na FKA (videti [41, 42, 107, 193, 211]). Stoga je razvoj efikasnih algoritama za računanje najvećih desno i levo invarijantnih fazi kvazi-uređenja i fazi ekvivalencija za dati fazi automat od izuzetne važnosti.

Prema [45, Teorema 4.2], za svaki fazi automat postoji najveća desno (odnosno, levo) invarijantna fazi ekvivalencija, dok prema [197, Teorema 4.3] za svaki fazi automat postoji najveće desno (odnosno, levo) invarijantno fazi kvazi-uređenje. Takođe, u istim radovima razvijeni su i polinomijalni algoritmi za njihovo računanje bazirani na nalaženju najveće fiksne tačke koja postoji na osnovu Kleeneove Teoreme o fiksnoj tački. Bez obzira na to, ti algoritmi nisu u mogućnosti da ih pronađu u svim slučajevima. Preciznije, u istim radovima pokazano je da se pomenuti algoritmi završavaju u konačnom broju koraka samo kada je struktura istinitosnih vrednosti lokalno konačna. Sa druge strane, kada struktura istinitosnih vrednosti nije lokalno konačna, pomenuti algoritmi se mogu, ali i ne moraju završiti u konačnom broju

koraka [44, 45, 197].

U ovoj Glavi prikazuju se algoritmi za računanje najvećih desno i levo invarijantnih fazi kvazi-uređenja, kao i najvećih desno i levo invarijantnih fazi ekvivalencija. Ovi algoritmi koriste dobro poznatu *tehniku usitnjenja particija*. Dobro je poznato da je problem nalaženja najveće bisimulacije ekvivalentan nalaženju najveće particije skupa stabilne u odnosu na datu relaciju [56, 57, 73]. Hopcroft [91] je prvi primenio tehniku usitnjenja particija u teoriji automata, i to u problemu minimizacije stanja DKA-a, da bi kasnije njegovi rezultati bili dalje objašnjeni i poboljšani u [82, 123]. Kanellakis i Smolka [114] rešili su ovaj problem u generalnom slučaju, da bi kasnije Paige i Tarjan [160] dodatno poboljšali njihov rezultat. Danas se tehnika usitnjenja particija koristi u mnogim oblastima računarskih nauka koje se bave teorijom grafova, stringovima, Boolovim matricama ili automatima (videti [84]).

Međutim, algoritmi za računanje najvećeg desno ili levo invarijantnog fazi kvazi-uređenja ili fazi ekvivalencije bazirani na tehnici usitnjenja particija pate od istog nedostatka kao i algoritmi bazirani na Kleeneovoj Teoremi o fiksnoj tački razvijeni u [44, 45, 197], a to je da se završavaju u konačnom broju koraka ako je struktura istinitosnih vrednosti lokalno konačna. Ovaj nedostatak može se prevazići ako se računaju desno i levo invarijantne krsip ekvivalencije na fazi automatu. Preciznije, ne samo da se algoritmi za računanje najvećih desno i levo invarijantnih krisp ekvivalencija na fazi automatu uvek završavaju u konačnom broju koraka, već se i izvršavaju brže od već razvijenih algoritama u [45, 197] za računanje najvećih desno i levo invarijantnih krisp ekvivalencija na fazi automatu.

Štaviše, ukoliko je struktura istinitosnih vrednosti fazi automata BL-algebra nad intervalom $[0, 1]$, dokazano je da se najveće desno ili levo invarijantno fazi kvazi-uređenje ili fazi ekvivalencija mogu odrediti preko granične vrednosti konvergentnih nizova konstruisanih u prezentovanim algoritmima. Poslednja tvrdnja sledi iz Sekcije 1.4, gde je opisano da u BL-algebrama na intervalu $[0, 1]$ postoji veza između neprekidnosti t-norme $\otimes$ i neprekidnosti $\otimes$ kao funkcije realne promenljive, što nam omogućuje da koristimo graničnu vrednost u uobičajenom smislu.

Rezultati iz ove glave objavljeni su u radu [146], a organizovani su na sledeći način. U Sekciji 3.1 dati su pojmovi desne i leve stabilnosti fazi relacija na fazi automatu, kao i definicije desno i levo invarijantnih fazi relacija. Potom se u Sekciji 3.2 proširuje pojam stabilnosti krisp relacija uveden u [160] na fazi relacije, i nalazi veza između stabilnosti fazi relacija u odnosu na druge fazi relacije i stabilnosti fazi relacija na fazi automatu (Lema 3.2). U istoj Sekciji se potom daju algoritmi za računanje najvećih desno i levo invarijantih fazi ekvivalencija (Algoritmi 1 i 2), kao i njihova formalna analiza. Kako se prikazani algoritmi ne moraju nužno završiti u konačnom broju koraka za proizvoljnu kompletnu reziduiranu mrežu, dokazujemo da nizovi fazi ekvivalencija generisani preko Algoritama 1 i 2 uvek konvergiraju u slučaju kada je struktura stepena istinitosti BL-algebra na realnom jediničnom intervalu $[0, 1]$, te da se najveća desno, odn. levo invarijantna fazi ekvivalencija dobija kao granična vrednost odgovarajućih nizova fazi ekvivalencija (Teorema 3.6). U Sekciji 3.3 uvodimo pojmove desne i leve stabilnosti fazi relacija u odnosu na druge fazi relacije, i dajemo algoritme za računanje najvećih desno i levo invarijantih fazi kvazi-uređenja na datom fazi automatu (Algoritmi 3 i 4). Preko Teoreme 3.11 dokazujemo da granična vrednost nizova fazi kvazi-uređenja generisanih preko Algoritama 3 i 4 postoji u istim uslovima kao i u Teoremi 3.6, te da je ona jednaka najvećem desno, odn. levo invarijantnom fazi kvazi-uređenju. Sekcija 3.4 posvećena je algoritmima za računanje najvećih desno i levo invarijantnih krisp ekvivalencija na datom fazi automatu. Ovi algoritmi pružaju ubrzanje u odnosu na odgovarajuće algoritme za

računanje najvećih desno i levo invarijantnih krisp ekvivalencija na datom fazi automatu razvijenih u [44, 45]. Naime, ukoliko sa m označimo broj simbola ulaznog alfabeta, a sa n broj stanja ulaznog fazi automata $\mathcal{A}$, tada algoritmi razvijeni u [44, 45] rade u $\mathcal{O}(mn^5)$ vremenskoj složenosti, dok Algoritam 5 dat u Sekciji 3.4 radi u $\mathcal{O}(mn^3)$ vremenskoj složenosti.

3.1 Desno i levo invarijantne fazi relacije

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat nad kompletnom reziduiranom mrežom $\mathcal{L}$ i alfabetom X. Za fazi relaciju $\varphi \in L^{A\times A}$ na A kaže se da je *desno stabilna na* $\mathcal{A}$ ako zadovoljava sledeći sistema fazi relacijskih nejednakosti:

$$\varphi \circ \delta_x \circ \varphi \leq \delta_x \circ \varphi, \quad \text{za svako } x \in X. \tag{3.1}$$

Slično, fazi relacija $\varphi \in L^{A\times A}$ je *levo stabilna na* $\mathcal{A}$ ako zadovoljava sledeći sistema fazi relacijskih nejednakosti:

$$\varphi \circ \delta_x \circ \varphi \leq \varphi \circ \delta_x, \quad \text{za svako } x \in X. \tag{3.2}$$

Dalje, za fazi relaciju $\varphi \in L^{A\times A}$ kaže se da je *desno invarijantna na* $\mathcal{A}$ ako je desno stabilna na $\mathcal{A}$ i ako je τ proširujuć u osnosu na φ^{-1}, odnosno, ako zadovoljava (3.1) i:

$$\varphi \circ \tau \leq \tau. \tag{3.3}$$

Analogno, fazi relacija $\varphi \in L^{A\times A}$ je *levo invarijantna na* $\mathcal{A}$ ako je levo stabilna na $\mathcal{A}$ i ako je σ proširujuć u osnosu na φ, odnosno, ako zadovoljava (3.2) i:

$$\sigma \circ \varphi \leq \sigma. \tag{3.4}$$

S obzirom da fazi relacija $\varphi \in L^{A\times A}$ zadovoljava (3.3) akko je:

$$\varphi \leq \tau / \tau = \pi^{fs},$$

zaključujemo da je desno stabilna fazi relacija na fazi automatu $\mathcal{A}$ desno invarijantna akko je sadržana u π^{fs}. Slično, s obzirom da fazi relacija $\varphi \in L^{A\times A}$ zadovoljava (3.4) akko je:

$$\varphi \leq \sigma \backslash \sigma = \pi^{is},$$

zaključujemo da je levo stabilna fazi relacija na fazi automatu $\mathcal{A}$ levo invarijantna akko je sadržana u π^{is}.

S obzirom da je svako fazi kvazi-uređenje refleksivno, lako se pokazuje da je fazi kvazi-uređenje $\varphi \in \mathcal{Q}(A)$ desno invarijantno akko zadovoljava sledeći sistem fazi relacijskih jednakosti:

$$\varphi \circ \delta_x \circ \varphi = \delta_x \circ \varphi, \quad \text{za svako } x \in X, \tag{3.5}$$
$$\varphi \circ \tau = \tau, \tag{3.6}$$

a da je fazi kvazi-uređenje $\varphi \in \mathcal{Q}(A)$ levo invarijantno akko zadovoljava sledeći sistem fazi relacijskih jednakosti:

$$\varphi \circ \delta_x \circ \varphi = \varphi \circ \delta_x, \quad \text{za svako } x \in X, \tag{3.7}$$
$$\sigma \circ \varphi = \sigma. \tag{3.8}$$

Primetimo da je i fazi ekvivalencija $\varphi \in \mathcal{E}(A)$ desno invarijantna akko zadovoljava sisteme (3.5) i (3.6), i da je $\varphi \in \mathcal{E}(A)$ levo invarijantna akko zadovoljava sisteme (3.7) i (3.8).

Krisp kvazi-uređenje (resp. krisp ekvivalencija) na A koje zadovoljava sisteme (3.5) i (3.6) naziva se *desno invarijantno kvazi-uređenje* (resp. *desno invarijantna ekvivalencija*) *na* $\mathcal{A}$, a kvazi-uređenje (resp. ekvivalencija) na A koje zadovoljava sisteme (3.7) i (3.8) naziva se *levo invarijantno kvazi-uređenje* (resp. *levo invarijantna ekvivalencija*) *na* $\mathcal{A}$. Primetimo da je (fazi) kvazi-uređenje na A ujedno i desno i levo invarijantno na $\mathcal{A}$ akko je

$$\delta_x \circ \varphi = \varphi \circ \delta_x, \quad \text{za svako } x \in X,$$

i tada se naziva *stabilno (fazi) kvazi-uređenje na* $\mathcal{A}$. Takođe, iz svih definicija izostavljamo fazi automat $\mathcal{A}$ ukoliko je jasan iz konteksta.

Takođe, za fazi relaciju $\varphi \in L^{A\times A}$ kaže se da je *slabo desno invarijantna* ako je:

$$\varphi \circ \tau_u \leq \tau_u, \quad \text{za svako } u \in X^*, \tag{3.9}$$

i dualno, φ je *slabo levo invarijantna* ako je:

$$\sigma_u \circ \varphi \leq \sigma_u, \quad \text{za svako } u \in X^*. \tag{3.10}$$

Lako je proveriti da je svaka desno invarijantna fazi relacija ujedno i slabo desno invarijantna, kao i da je svaka levo invarijantna fazi relacija ujedno i slabo levo invarijantna. Primetimo da, ukoliko je φ refleksivna, tada φ zadovoljava (3.9) ako i samo ako zadovoljava:

$$\varphi \circ \tau_u = \tau_u, \quad \text{za svako } u \in X^*, \tag{3.11}$$

i φ zadovoljava (3.10) ako i samo ako zadovoljava:

$$\sigma_u \circ \varphi = \sigma_u, \quad \text{za svako } u \in X^*. \tag{3.12}$$

Slabo desno invarijantne i slabo levo invarijantne fazi relacije uvedene su u [197], gde su korišćene u redukciji stanja fazi automata. One daju fazi automate sa manjim brojem stanja u odnosu na desno i levo invarijantne fazi relacije, ali ih je teže izračunati (naime, algoritmi za njihovo računanje rade u eksponencijalnoj vremenskoj složenosti). Slabo desno invarijantne i slabo levo invarijantne fazi kvazi-uređenja i fazi ekvivalencije u uskoj su vezi sa slabim simulacijama i slabim bisimulacijama za fazi automate, koje su izučavane u [107].

Naredna Teorema predstavlja verziju rezultata iz [197] i daje karakterizaciju skupa svih desno invarijantnih fazi kvazi-uređenja na datom FKA-u.

Teorema 3.1. *Za svaki FKA* $\mathcal{A}$ *važi:*

(1) *Skup svih desno stabilnih fazi kvazi uređenja* $\mathcal{Q}^{rs}(\mathcal{A})$ *formira kompletnu mrežu, i predstavlja kompletnu supremum-polumrežu mreže* $\mathcal{Q}(A)$ *svih fazi kvazi-uređenja na* A*,*

(2) *Skup svih desno invarijantnih fazi kvazi-uređenja* $\mathcal{Q}^{ri}(\mathcal{A})$ *formira glavni ideal mreže* $\mathcal{Q}^{rs}(\mathcal{A})$*.*

3.2 Računanje najvećih desno i levo invarijantnih fazi ekvivalencija

Dobro poznati algoritam Paige i Tarjana zasniva se na pojmu *stabilnosti* blokova i particija (videti [1, 48, 160]). Zato najpre dajemo definiciju stabilnosti u fazi slučaju.

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA i $\varphi, \phi \in \mathcal{E}(A)$ dve fazi ekvivalencije na A. Kažemo da je φ *stabilna* u odnosu na ϕ ako važi:

$$\varphi \leq \bigwedge_{x \in X} (\delta_x \circ \phi^a) | (\delta_x \circ \phi^a), \quad \text{za svako } a \in A. \tag{3.13}$$

Naredna Lema daje vezu između stabilnih fazi ekvivalencija na fazi automatu i stabilnih fazi ekvivalencija u odnosu na druge fazi ekvivalencije.

Lema 3.2. *Neka je $\mathcal{A}$ FKA i $\varphi \in \mathcal{E}(A)$. Tada je φ stabilna u odnosu na samu sebe ako je desno stabilna na $\mathcal{A}$.*

Dokaz. Neka je $\varphi \in \mathcal{E}(A)$ desno stabilna na $\mathcal{A}$. Tada se (3.1) može na ekvivalentan način zapisati kao:

$$\bigvee_{b \in A} \varphi(c, b) \otimes (\delta_x \circ \varphi)(b, a) \leq (\delta_x \circ \varphi)(c, a), \quad \text{za svako } x \in X \text{ i } a, c \in A,$$

što je dalje ekvivalentno sa:

$$\varphi(c, b) \otimes (\delta_x \circ \varphi)(b, a) \leq (\delta_x \circ \varphi)(c, a), \quad \text{za svako } x \in X \text{ i } a, b, c \in A,$$

a nadalje, prema svojstvu adjunkcije (1.11), ekvivalentno sa:

$$\varphi(c, b) \leq (\delta_x \circ \varphi)(b, a) \to (\delta_x \circ \varphi)(c, a), \quad \text{za svako } x \in X \text{ i } a, b, c \in A,$$

odnosno:

$$\varphi(c, b) \leq (\delta_x \circ \varphi^a)(b) \to (\delta_x \circ \varphi^a)(c), \quad \text{za svako } x \in X \text{ i } a, b \in A. \tag{3.14}$$

Takođe, s obzirom da je φ simetrična fazi relacija, sledi

$$\varphi(b, c) \otimes (\delta_x \circ \varphi)(b, a) \leq (\delta_x \circ \varphi)(c, a), \quad \text{za svako } x \in X \text{ i } a, b, c \in A,$$

što je prema svojstvu adjunkcije (1.11) ekvivalentno sa:

$$\varphi(b, c) \leq (\delta_x \circ \varphi^a)(b) \to (\delta_x \circ \varphi^a)(c), \quad \text{za svako } x \in X \text{ i } a, b \in A. \tag{3.15}$$

Na osnovu (3.14) i (3.15) sledi da φ zadovoljava (3.13). □

Naredna Lema biće od izuzetnog značaja za dalje izlaganje.

Lema 3.3. *Neka je $\mathcal{A}$ FKA i $\varphi, \phi \in \mathcal{E}(A)$ fazi ekvivalencije na A takva da je φ usitnjenje od ϕ. Tada je φ stabilna u odnosu na ϕ ako je φ desno stabilna na $\mathcal{A}$.*

Dokaz. Ako je $\varphi \in \mathcal{E}(A)$ desno stabilna, tada važi (3.1), te je stoga:

$$\varphi \circ \delta_x \circ \varphi \circ \phi \leq \delta_x \circ \varphi \circ \phi, \quad \text{za svako } x \in X.$$

Iz Leme 1.24 sledi:

$$\varphi \circ \delta_x \circ \phi \leq \delta_x \circ \phi, \quad \text{za svako } x \in X,$$

što je ekvivalentno sa:

$$\varphi(b, c) \otimes (\delta_x \circ \phi)(c, a) \leq (\delta_x \circ \phi)(b, a), \quad \text{za svako } x \in X \text{ i } a, b, c \in A. \tag{3.16}$$

Koristeći svojstvo adjunkcije (1.11) uočavamo da je gornja nejednakost ekvivalentna sa:

$$\varphi(b,c) \leq (\delta_x \circ \phi^a)(c) \to (\delta_x \circ \phi^a)(b), \quad \text{za svako } x \in X \text{ i } a,b,c \in A, \tag{3.17}$$

Takođe, s obzirom da je φ simetrična fazi relacija, iz (3.16) sledi

$$\varphi(c,b) \otimes (\delta_x \circ \phi)(c,a) \leq (\delta_x \circ \phi)(b,a), \quad \text{za svako } x \in X \text{ i } a,b,c \in A,$$

a što je potom na osnovu svojstva adjunkcij (1.11) ekvivalentno sa:

$$\varphi(c,b) \leq (\delta_x \circ \phi^a)(c) \to (\delta_x \circ \phi^a)(b), \quad \text{za svako } x \in X \text{ i } a,b,c \in A, \tag{3.18}$$

te (3.13) sledi iz (3.17) i (3.18). □

Za fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$, označimo sa $\mathcal{L}(\delta, \tau)$ podalgebru od $\mathcal{L}$ generisanu svim vrednostima od δ i τ.

Naredna Teorema pruža metod za računanje najveće desno invarijantne fazi ekvivalencije na datom fazi automatu. Najpre, primetimo da $\varphi \in \mathcal{E}(A)$ zadovoljava (3.3) akko $\varphi \leq \tau | \tau = \pi^{fb}$.

Teorema 3.4. *Neka je $\mathcal{A}$ FKA. Definišimo induktivno nizove $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ fazi relacija na A na sledeći način: Za $k = 1$, odaberimo proizvoljno $a \in A$ i stavimo da je:*

$$\phi_1 = U, \tag{3.19}$$

$$\varphi_1 = \pi^{fb} \wedge \bigwedge_{x\in X} (\delta_x \circ \phi_1^a) | (\delta_x \circ \phi_1^a), \tag{3.20}$$

Pretpostavimo da su u koraku k izračunate fazi relacije ϕ_k i φ_k. U sledećem koraku, ako je $\phi_k \neq \varphi_k$, odaberimo proizvoljno $a \in A$ tako da je $\phi_k^a \neq \varphi_k^a$, i izračunajmo skup $B = \{b \in A | \varphi_k^a(b) \neq 1\}$ (odnosno, krisp podskup skupa A čiji elementi jasno ne pripadaju klasi ekvivalencije φ_k^a). Potom izračunajmo fazi relacije ϕ_{k+1} i φ_{k+1} na sledeći način:

$$\phi_{k+1} = \phi_k \wedge (\varphi_k^a | \varphi_k^a), \tag{3.21}$$

$$\varphi_{k+1} = \varphi_k \wedge \left(\bigwedge_{x\in X} (\delta_x \circ \varphi_k^a) | (\delta_x \circ \varphi_k^a) \right) \wedge \left(\bigwedge_{x\in X} \bigwedge_{b\in B} (\delta_x \circ \phi_{k+1}^b) | (\delta_x \circ \phi_{k+1}^b) \right), \tag{3.22}$$

Inače, postavimo da je $\phi_{k+1} = \phi_k$ i $\varphi_{k+1} = \varphi_k$. Tada važi sledeće:

(a) Nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ su opadajući;

(b) $\phi_k \in \mathcal{E}(A)$ i $\varphi_k \in \mathcal{E}(A)$ (čime se opravdavaju gore upotrebljene oznake za fazi klase ekvivalencije), za svako $k \in \mathbb{N}$;

(c) φ_k je usitnjenje od ϕ_k, za svako $k \in \mathbb{N}$;

(d) φ_k je stabilna u odnosu na ϕ_k, za svako $k \in \mathbb{N}$;

(e) Za svako $k \in \mathbb{N}$ i $x \in X$ važi:

$$\varphi_k \circ \delta_x \circ \phi_k \leq \delta_x \circ \phi_k; \tag{3.23}$$

(f) Ako je $\phi_s = \varphi_s$, za neko $s \in \mathbb{N}$, tada je $\phi_s = \varphi_s$ najveća desno invarijantna fazi ekvivalencija na A;

(g) Ako $\mathcal{L}(\delta, \tau)$ zadovoljava UOL, tada se nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ stabilizuju, odnosno postoji $s \in \mathbb{N}$ tako da je $\phi_s = \varphi_s$.

Dokaz. (a) Sledi direktno iz definicija nizova $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$.

(b) Na osnovu Lema 1.23 i 1.27, sledi $\phi_k \in \mathcal{E}(A)$ i $\varphi_k \in \mathcal{E}(A)$, za svako $k \in \mathbb{N}$.

(c) Dokazujemo da je $\varphi_k \leq \phi_k$ za svako $k \in \mathbb{N}$ indukcijom po k. Za $k = 1$ na osnovu (3.19) važi $\varphi_1 \leq U = \phi_1$. Pretpostavimo da je $\varphi_m \leq \phi_m$ za neko $k = m$. Prema Lemi 1.26 imamo da je $\varphi_m \leq \varphi_m^a|\varphi_m^a$, pri čemu je $a \in A$ odabrano u koraku m tako da važi $\varphi_m^a \neq \phi_m^a$. Na osnovu indukcijske pretpostavke sledi $\varphi_m \leq \phi_m \wedge (\varphi_m^a|\varphi_m^a) = \phi_{m+1}$. Na osnovu dela (a), niz $\{\varphi_k\}_{k\in\mathbb{N}}$ je opadajuć, te sledi $\varphi_{m+1} \leq \varphi_m \leq \phi_{m+1}$, što je i trebalo dokazati.

(d) Dokazujemo da važi

$$\varphi_k \leq \bigwedge_{x\in X}\bigwedge_{a\in A}(\delta_x \circ \phi_k^a)|(\delta_x \circ \phi_k^a), \tag{3.24}$$

za svako $k \in \mathbb{N}$ indukcijom po k. Najpre, primetimo da je $\phi_1 = U$, te su sve fazi klase ekvivalencije od ϕ_1 jednake fazi podskupu od A čiji su svi elementi jednaki 1. Tada direktno iz definicije (3.24) sledi da za svako $a \in A$ važi:

$$\varphi_1 \leq \bigwedge_{x\in X}(\delta_x \circ \phi_1^a)|(\delta_x \circ \phi_1^a),$$

odakle (3.38) direktno sledi. Pretpostavimo sada da (3.24) važi za neko $k = m$, te dokažimo da važi i za $k = m + 1$. Najpre, primetimo da direktno iz definicije (3.22) sledi:

$$\varphi_{m+1} \leq \bigwedge_{x\in X}(\delta_x \circ \phi_{m+1}^c)|(\delta_x \circ \phi_{m+1}^c), \quad \text{za svako } c \in B, \tag{3.25}$$

gde su, prema konstrukciji, $a \in A$ takvo da je $\varphi_m^a \neq \phi_m^a$ i $B = \{b \in A|\varphi_k^a(b) \neq 1\}$. Dokažimo sada da je $\phi_{m+1}^c = \varphi_m^a$ za svako $c \in A - B$. Zaista, za svako $d \in A$ imamo da na osnovu (3.21) važi:

$$\phi_{m+1}^c(d) = \phi_{m+1}(d,c) = \phi_m(d,c) \wedge (\varphi_m^a|\varphi_m^a)(d,c) = \phi_m^c(d) \wedge (\varphi_m^a(c) \leftrightarrow \varphi_m^a(d)).$$

S obzirom da je $C = A - B$, dalje dobijamo da je

$$\phi_{m+1}^c(d) = \phi_m^c(d) \wedge (1 \leftrightarrow \varphi_m^a(d)) = \phi_m^c(d) \wedge \varphi_m^a(d) = \varphi_m^a(d), \tag{3.26}$$

pri čemu poslednja jednakost sledi na osnovu dela (c). Sada, s obzirom da takođe iz definicije (3.22) važi:

$$\varphi_{m+1} \leq \bigwedge_{x\in X}(\delta_x \circ \varphi_m^a)|(\delta_x \circ \varphi_m^a),$$

tada na osnovu (3.26) važi i:

$$\varphi_{m+1} \leq \bigwedge_{x\in X}(\delta_x \circ \phi_{m+1}^c)|(\delta_x \circ \phi_{m+1}^c), \quad \text{za svako } c \in A - B, \tag{3.27}$$

te na osnovu (3.25) i (3.27) dobijamo da nejednakost (3.24) važi za svako $k = m + 1$, što je i trebalo dokazati.

(e) Sledi direktno iz (d).

(f) Ako je $\varphi_s = \phi_s$, za neko $s \in \mathbb{N}$, tada je na osnovu (3.23) φ_s desno stabilna na $\mathcal{A}$, dok je na osnovu (3.20) $\varphi_s \leq \varphi_1 \leq \pi^{fb}$, što znači da je φ_s desno invarijantna fazi ekvivalencija na A. Dokažimo sada da je φ_k ujedno i najveća desno invarijantna fazi ekvivalencija na A. Da bismo to uradili, dokazaćemo da je $\psi \leq \varphi_k$ za svaku desno invarijantnu fazi ekvivalenciju ψ na A i $k \in \mathbb{N}$ indukcijom po k.
U slučaju $k = 1$, s obzirom da su ψ i ϕ_1 fazi ekvivalencije na A takva da je $\psi \leq$

$U = \phi_1$, prema Lemi 3.3 dobijamo da je φ stabilna u odnosu na ϕ_1, što znači da je $\psi \leq \bigwedge_{x\in X}(\delta_x \circ \phi_1^a)|(\delta_x \circ \phi_1^a)$ za svako $a \in A$. Takođe važi i $\psi \leq \tau|\tau = \pi^{fb}$, što znači da je $\psi \leq \pi^{fb} \wedge \bigwedge_{x\in X}(\delta \circ \phi_1^a)|(\delta \circ \phi_1^a) = \varphi_1$.
Pretpostavimo da je

$$\psi \leq \varphi_m, \quad \text{za neko } k = m. \tag{3.28}$$

Tada iz Leme 3.3 sledi

$$\psi \leq \bigwedge_{x\in X}(\delta_x \circ \varphi_m^a)|(\delta_x \circ \varphi_m^a), \quad \text{za svako } a \in A. \tag{3.29}$$

Najzad, iz (3.28) i (b) sledi $\psi \leq \phi_m$, a na osnovu Leme 1.26 i $\psi \leq \phi_{m+1}$. Potom, iz Leme 3.9 sledi i

$$\psi \leq \bigwedge_{x\in X}(\delta_x \circ \phi_{m+1}^a)|(\delta_x \circ \phi_{m+1}^a), \quad \text{za svako } a \in A. \tag{3.30}$$

Na osnovu (3.28), (3.29) i (3.30) sledi $\psi \leq \varphi_{m+1}$, što je i trebalo dokazati.

(g) Ukoliko $\mathcal{L}(\delta, \tau)$ zadovoljava UOL, tada se fazi relacija ϕ_k može posmatrati kao matrica sa elementima u $\mathcal{L}(\delta, \tau)$, za svako $k \in \mathbb{N}$. Za svaki par $(a, b) \in A \times A$, elementi na poziciji (a, b) u ovim matricama formiraju nerastući niz $\{\phi_k(a, b)\}_{k\in\mathbb{N}}$ elemenata iz $\mathcal{L}(\delta, \tau)$. Prema pretpostavci, svi ovakvi nizovi se stabilizuju, odnosno postoji $s \in \mathbb{N}$ takav da se svi ovakvi nizovi posle s članova stabilizuju. To znači da se i niz $\{\phi_k(a, b)\}_{k\in\mathbb{N}}$ fazi ekvivalencija takođe stabilizuje nakon s članova, što znači $\phi_s = \phi_{s+1}$.
Dokažimo sada da iz $\phi_s = \phi_{s+1}$ sledi $\phi_s = \varphi_s$. Iz $\phi_s = \phi_{s+1}$ na osnovu (3.21) imamo da važi $\phi_s \leq (\varphi_s^a|\varphi_s^a)$. Stoga je i $\phi_s^a \leq \varphi_s^a$, a s obzirom da je $\varphi_s^a \leq \phi_s^a$, dobijamo da je $\phi_s^a = \varphi_s^a$. To znači da u koraku s konstrukcije nizova $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ ne postoji fazi klasa ekvivalencije ϕ_s^a od ϕ_s tako da je $\phi_s^a \neq \varphi_s^a$, ili ekvivalentno $\phi_s = \varphi_s$. □

Teorema 3.4 može se transformisati u proceduru za računanje najveće desno invarijantne fazi ekvivalencije na FKA-u $\mathcal{A}$, i data je Algoritmom 1.

Algoritam 1 Računanje najveće desno invarijantne fazi ekvivalencije na fazi automatu

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad alfabetom X i kompletnom reziduiranom mrežom $\mathcal{L}$.
Izlaz: Najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$.

1: $\phi \leftarrow U$
2: $\varphi \leftarrow \pi^{fb} \wedge (\bigwedge_{x\in X}(\delta_x \circ \phi^a)|(\delta_x \circ \phi^a))$, za proizvoljno $a \in A$
3: **while** $\phi \neq \varphi$ **do**
4: Odaberimo proizvoljno $a \in A$ tako da je $\phi^a \neq \varphi^a$
5: Izračunati $B = \{b \in A|\varphi^a(b) \neq 1\}$
6: $\phi_1 \leftarrow \phi \wedge (\varphi^a|\varphi^a)$
7: $\varphi_1 \leftarrow \varphi \wedge \big(\bigwedge_{x\in X}(\delta_x \circ \varphi^a)|(\delta_x \circ \varphi^a)\big) \wedge \big(\bigwedge_{x\in X}\bigwedge_{b\in B}(\delta_x \circ \phi_1^b)|(\delta_x \circ \phi_1^b)\big)$
8: $\phi \leftarrow \phi_1$, $\varphi \leftarrow \varphi_1$
9: **end while**
10: **return** φ

Odredimo vremensku složenost Algoritma 1. Neka je $|A| = n$ i $|X| = m$, i neka su $c_\vee$, $c_\wedge$, $c_\otimes$ i $c_\to$, respektivno, vremena izvršenja operacija $\vee$, $\wedge$, $\otimes$, $\to$ u $\mathcal{L}$. Kada je $\mathcal{L}$

standardna proizvod, standardna Gödelova ili standardna Łukasiewiczeva algebra, tada je $c_\vee = c_\wedge = c_\otimes = c_\to = 1$.

Očigledno, Korak 1 izvršava se u $\mathcal{O}(n^2)$ vremenu. Da bismo izračunali φ u Koraku 2, najpre je potrebno izračunati fazi relaciju π^{fb}, što se vrši u $O(n^2(c_\to + c_\wedge))$ vremenu, potom kompoziciju fazi relacije δ_x i fazi podskupa ϕ^a, za neko $a \in A$, što se vrši u $O(n^2(c_\otimes + c_\vee))$ vremenu. Na kraju, računanje izraza $\bigwedge_{x\in X}(\delta_x \circ \phi^a)|(\delta_x \circ \phi^a)$ vrši se u $O(mn^2(c_\otimes + c_\vee) + mn^2(c_\to + c_\wedge))$ vremenu. Stoga je ukupno vreme izvršenja Koraka 2 jednako $O(mn^2(c_\otimes + c_\vee + c_\to + c_\wedge))$.

Vreme potrebno da se nađe $a \in A$ u Koraku 4 takvo da je $\phi^a \neq \varphi^a$ proporcionalno je sa $O(n^2)$, dok Korak 5 zahteva $\mathcal{O}(n)$ vremena.

Računanje fazi relacije ϕ_1 u Koraku 6 preko izraza (3.21) zahteva $O(n^2(c_\to + c_\wedge))$ vremena, dok računanje fazi relacije φ_1 u Koraku 7 preko izraza (3.22) zahteva $O(mn^3(c_\otimes + c_\vee + c_\to + c_\wedge))$ vremena.

Najteže je odrediti broj iteracija koje zahteva petlja kroz Korake 3–9, u slučajevima kada je broj takvih iteracija konačan. Posmatrajmo samo elemente niza $\{\phi_k\}_{k\in\mathbb{N}}$ s obzirom da oba niza $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$, generisanih preko formula (3.19) – (3.22), imaju isti broj elemenata ako su konačni. Kao što je već napomenuto, svaka fazi relacija ϕ_k predstavlja matricu sa elementima iz $\mathcal{L}(\delta, \tau)$. Ukoliko $\mathcal{L}(\delta, \tau)$ zadovoljava UOL, tada je niz $\{\phi_k\}_{k\in\mathbb{N}}$ konačan, pa postoji broj $l \in \mathbb{N}$ takav da je broj različitih elemenata u svakom od tog niza manji ili jednak od l. Takođe, niz $\{\phi_k\}_{k\in\mathbb{N}}$ je opadajući, svaki element može da promeni vrednost najviše $l-1$ puta, te je broj promena manji ili jednak $(l-1)(n^2-n)$ (elementi na glavnoj dijagonali matrice moraju biti jednaki 1). Stoga se petlja završava najviš nakon $(l-1)(n^2-n)+2$ koraka (izmene se dešavaju između drugog i pretposlednjeg koraku).

Sumirajući, ukupno vreme izrvšenja Algortma 1 jednako je $O(lmn^5(c_\otimes + c_\vee + c_\to + c_\wedge))$, odnosno, vreme izvršenja algoritma je polinomijalno.

Napominjemo da možemo pretpostaviti da je l broj elemenata podalgebre $\mathcal{L}(\delta, \tau)$ u slučaju kada je podalgebra konačna. Posebno, kada je $\mathcal{L}$ standardna Gödelova algebra, tada su jedine vrednosti koje mogu uzeti elementi fazi relacija iz familije $\{\phi_k\}_{k\in\mathbb{N}}$ jesu 1 i one vrednosti koje uzimaju elementi fazi relacija iz familije $\{\delta_x\}_{x\in X}$, i možemo pretpostaviti da je $l = j+1$, pri čemu je j broj različitih vrednosti koje uzimaju elementi fazi relacija iz familije $\{\delta_x\}_{x\in X}$. U ovoj strukturi, Algoritam 1 završava se u $j(n^2-n)+2$ koraka, ili najviše posle $(m+1)n^2(n^2-n)+2$ koraka, s obzirom da je $j \leq mn^2+n^2$, ili drugim rečima u $O(jmn^5)$ vremenu, što je proporcionalno sa $O(m^2n^7)$.

Na kraju, ukoliko je $\mathcal{L}$ Boolova struktura, tada je $l = 2$, pa se Algoritam završava posle najviše n^2-n+2 koraka, što znači da je njegova kompleksnost u tom slučaju jednaka $O(mn^5)$.

Narednim primerom demonstrira se rad Algoritma 1.

Primer 3.5. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA nad $\mathcal{L}$ i X, pri čemu je $\mathcal{L}$ standardna proizvod (Gougenova) algebra, $X = \{x\}$, $A = \{a_1, a_2, a_3, a_4, a_5\}$, a σ, δ_x i τ zadate sledećim matricama:

$$\sigma = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \delta_x = \begin{bmatrix} 0 & 1 & 0 & 0.6 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0.6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0.4 & 0.8 \\ 0.6 & 0.2 & 0 & 0 & 0 \end{bmatrix}, \quad \tau = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

U prvom koraku, Algoritam 1 daje sledeće matrice:

$$\phi_1 = U = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$\varphi_1 = \tau | \tau \wedge (\delta_x \circ \phi_1^a) | (\delta_x \circ \phi_1^a) = \begin{bmatrix} 1 & 1 & 0 & 0.8 & 0 \\ 1 & 1 & 0 & 0.8 & 0 \\ 0 & 0 & 1 & 0 & 0.6 \\ 0.8 & 0.8 & 0 & 1 & 0 \\ 0 & 0 & 0.6 & 0 & 1 \end{bmatrix}.$$

Vidimo da fazi ekvivalencija ϕ_1 sadrži jednu klasu ekvivalencije, dok fazi ekvivalencija φ_1 sadrži četiri klase ekvivalencije koje su pritom sve različite od klase ekvivalencije od ϕ_1. Stoga, odaberimo $a_1 \in A$ (za koje je $\phi_1^{a_1} \neq \varphi_1^{a_1}$. Tada je $B = \{b \in A | \varphi_1^{a_1}(b) \neq 1\} = \{a_3, a_4, a_5\}$. U sledećem koraku, Algoritam 1 daje sledeće matrice:

$$\phi_2 = \phi_1 \wedge (\varphi_1^{a_1} | \varphi_1^{a_1}) = \begin{bmatrix} 1 & 1 & 0 & 0.8 & 0 \\ 1 & 1 & 0 & 0.8 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0.8 & 0.8 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$\varphi_2 = \varphi_1 \wedge (\delta_x \circ \varphi_1^{a_1}) | (\delta_x \circ \varphi_1^{a_1}) \wedge \left(\bigwedge_{b \in B} (\delta_x \circ \phi_2^b) | (\delta_x \circ \phi_2^b) \right) = \begin{bmatrix} 1 & 0.8 & 0 & 0 & 0 \\ 0.8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.6 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.6 & 0 & 1 \end{bmatrix}.$$

Sada fazi ekvivalencija ϕ_2 sadrži četiri klase ekvivalencije, dok fazi ekvivalencija φ_2 sadrži pet klasa ekvivalencije i koje su pritom sve međusobno različite. Odaberimo ponovo $a_1 \in A$ jer važi $\phi_2^{a_1} \neq \varphi_2^{a_1}$. Sada je $B = \{b \in A | \varphi_2^{a_1}(b) \neq 1\} = \{a_2, a_3, a_4, a_5\}$, te u sledećem koraku dobijamo matrice:

$$\phi_3 = \phi_2 \wedge (\varphi_2^{a_1} | \varphi_2^{a_1}) = \begin{bmatrix} 1 & 0.8 & 0 & 0 & 0 \\ 0.8 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$\varphi_3 = \varphi_2 \wedge (\delta_x \circ \varphi_2^{a_1}) | (\delta_x \circ \varphi_2^{a_1}) \wedge \left(\bigwedge_{b \in B} (\delta_x \circ \phi_3^b) | (\delta_x \circ \phi_3^b) \right) = \begin{bmatrix} 1 & 0.6 & 0 & 0 & 0 \\ 0.6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.48 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.48 & 0 & 1 \end{bmatrix}.$$

Nastavljajući u istom maniru dobijamo:

$$\phi_4 = \begin{bmatrix} 1 & 0.6 & 0 & 0 & 0 \\ 0.6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad \varphi_4 = \begin{bmatrix} 1 & 0.6 & 0 & 0 & 0 \\ 0.6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.36 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.36 & 0 & 1 \end{bmatrix},$$

$\phi_5 = \varphi_5 = \varphi_4$.

Dakle, Algoritam 1 završava se nakon 5 koraka kada dobijamo da je $\phi_5 = \varphi_5$ najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$.

Na osnovu Teoreme 3.4 (g), Algoritam 1 završava se u konačnom broju koraka ako je $\mathcal{L}$ lokalno konačna. Prethodni primer pokazuje da se Algoritam 1 može takođe završiti u konačnom broju koraka ako $\mathcal{L}$ nije lokano konačna, ali to inače ne mora biti slučaj. Postavlja se pitanje kako odrediti najveću desno invarijantnu fazi ekvivalenciju u slučajevima kada Algoritam 1 daje beskonačne nizove fazi ekvivalencija. Sada pokazujemo da su ovi nizovi fazi ekvivalencija konvergentni u slučaju kada je $\mathcal{L} = ([0,1], \min, \max, \otimes, \to, 0, 1)$ BL-algebra, i da je tada najveća desno invarijantna fazi ekvivalencija za dati fazi automat jednaka graničnoj vrednosti tih konvergentnih nizova. To znači da se naredna Teorema može iskoristiti za dobijanje najveće desno invarijantna fazi ekvivalencija za dati fazi automat kada se Algoritam 1 ne završava u konačnom broju koraka. Podsetimo se da su standardna prozivod, standardna Gödelova i standardna Łukasiewiczeva algebra primeri BL-algebre.

Teorema 3.6. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA nad $\mathcal{L}$ i X, pri čemu je $\mathcal{L}$ BL-algebra nad realnim jediničnim intervalom $[0,1]$. Tada su nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$, definisani preko* (3.19) – (3.22), *konvergentni. Ako označimo $\lim_{n\to\infty} \phi_n = \lim_{n\to\infty} \varphi_n = \varphi$, tada je φ najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$.*

Dokaz. Najpre primetimo da su nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ monotoni, s obzirom da je u Teoremi 3.4 (a) dokazano da su nizovi opadajući. Takođe, s obzirom da su fazi relacije ϕ_k i φ_k refleksivne za svako $k \in \mathbb{N}$, imamo da su oba niza ograničena, s obzirom da za svako $k \in \mathbb{N}$ važi:

$$\mathbf{I} \leq Q_k \leq \mathcal{U}, \quad \mathbf{I} \leq R_k \leq \mathcal{U},$$

gde je, kao i obično, sa $\mathbf{I}$ označena identička fazi relacija a sa $\mathcal{U}$ univerzalna fazi relacija.

S obzirom da svaki ograničeni monotoni niz konvergira u skupu realnih brojeva, zaključujemo da su nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ konvergentni. Štaviše, na osnovu definicije nizova direktno sledi da je $\lim_{n\to\infty} \phi_n = \lim_{n\to\infty} \varphi_n = \varphi$.

Dalje, dokažimo da je φ desno invarijantna fazi ekvivalencija na $\mathcal{A}$. Najpre, s obizorom da je $\{\varphi_k\}_{k\in\mathbb{N}}$ opadajući niz, sledi $\varphi \leq \varphi_k$ za svako $k \in \mathbb{N}$, pa je i $\varphi \leq \varphi_1 \leq \pi^{fb}$. Dakle, φ zadovoljava (3.3).

Da bismo pokazali da je φ desno stabila, koristimo činjenicu da je t-norma $\otimes$ neprekidna u BL-algebrama (videti Teoremu 1.17). Primenjujući (3.23) dobijamo da za svako $x \in X$ važi:

$$\lim_{n\to\infty} (\varphi_n \circ \delta_x \circ \phi_n) \leq \lim_{n\to\infty} (\delta_x \circ \phi_n),$$

odnosno:

$$\begin{aligned}\varphi \circ \delta_x \circ \varphi &= (\lim_{n\to\infty} \varphi_n) \circ \delta_x \circ (\lim_{n\to\infty} \phi_n) = \lim_{n\to\infty} (\varphi_n \circ \delta_x \circ \phi_n) \\ &\leq \lim_{n\to\infty} (\delta_x \circ \phi_n) = \delta_x \circ (\lim_{n\to\infty} \phi_n) \\ &= \delta_x \circ \varphi.\end{aligned}$$

Dakle, φ je desno invarijantna. Takođe, φ je refleksivna (sve fazi relacije φ_n su refleskivne), a takođe važi i:

$$\varphi^{-1} = \left(\lim_{n\to\infty} \varphi_n\right)^{-1} = \lim_{n\to\infty} (\varphi_n)^{-1} = \lim_{n\to\infty} \varphi_n = \varphi,$$

kao i:

$$\varphi \circ \varphi = \left(\lim_{n\to\infty} \varphi_n\right) \circ \left(\lim_{n\to\infty} \varphi_n\right) = \lim_{n\to\infty} (\varphi_n \circ \varphi_n) = \lim_{n\to\infty} \varphi_n = \varphi,$$

što redom znači da je φ simetrična i tranzitivna, odnosno, da je φ fazi ekvivalencija.

U Teoremi 3.4 (g) dokazali smo da za svaku desno invarijantnu fazi ekvivalenciju ψ važi $\psi \leq \varphi_n$ za svako $n \in \mathbb{N}$, te je stoga $\psi \leq \lim_{n\in\mathbb{N}} \varphi_n = \varphi$, odakle sledi da je φ najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$. □

Primer 3.7. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA nad $\mathcal{L}$ i X, pri čemu je $\mathcal{L}$ standardna proizvod algebra, $X = \{x, y\}$, $A = \{a_1, a_2, a_3\}$, kao i:

$$\sigma = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}, \quad \tau = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \quad \delta_x = \begin{bmatrix}0 & 0.8 & 0.5 & 0\\0 & 0.5 & 0 & 0\\0 & 0 & 0.4 & 0\\0 & 0 & 0 & 0\end{bmatrix}, \quad \delta_y = \begin{bmatrix}0 & 0 & 0 & 0\\0 & 0.3 & 0 & 0.5\\0 & 0 & 0.3 & 0.4\\0 & 0 & 0 & 0\end{bmatrix}.$$

Algoritam 1 daje sledeće matrice:

$$\phi_1 = \begin{bmatrix}1 & 1 & 1 & 1\\1 & 1 & 1 & 1\\1 & 1 & 1 & 1\\1 & 1 & 1 & 1\end{bmatrix}, \quad \varphi_1 = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0.8 & 0\\0 & 0.8 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix},$$

$$\phi_2 = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 1 & 1\\0 & 1 & 1 & 1\\0 & 1 & 1 & 1\end{bmatrix}, \quad \varphi_2 = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0.8 & 0\\0 & 0.8 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix},$$

$$\phi_k = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0.8^{k-2} & 0\\0 & 0.8^{k-2} & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}, \quad \varphi_k = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0.8^{k-1} & 0\\0 & 0.8^{k-1} & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}, \quad \text{za svako } k \geq 3.$$

Dakle, $\phi_k \neq \varphi_k$, za svako $k \in \mathbb{N}$, dakle nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ su beskonačni. Sa druge strane, na osnovu Teoreme 3.6 imamo da je

$$\lim_{k\to\infty} \phi_k = \lim_{k\to\infty} \varphi_k = \varphi = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix},$$

te imamo da je φ najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$.

Za fazi automat $\mathcal{A}$ nad $\mathcal{L}$ i X analogno se može računati i najveća levo invarijantna fazi ekvivalencija na $\mathcal{A}$, i metod za njeno računanje dat je Algortimom 2. Vremenska složenost ovog algoritma identična je vremenskoj složenosti Algoritma 1, a završava se kada algebra $\mathcal{L}(\sigma, \delta)$ zadovoljava UOL. Štaviše, rezultat analogan Teoremi 3.6 važi i za nizove generisanih preko Algoritma 2.

Algoritam 2 Računanje najveće levo invarijantne fazi ekvivalencije na fazi automatu

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad alfabetom X i kompletnom reziduiranom mrežom $\mathcal{L}$.

Izlaz: Najveća levo invarijantna fazi ekvivalencija na $\mathcal{A}$.

1: $\phi \leftarrow U$
2: $\varphi \leftarrow \pi^{fb} \wedge \big(\bigwedge_{x\in X}(\phi^a \circ \delta_x)|(\phi^a \circ \delta_x)\big)$, za proizvoljno $a \in A$
3: **while** $F \neq E$ **do**
4: Odaberimo proizvoljno $a \in A$ tako da je $\phi^a \neq \varphi^a$
5: Izračunati $B = \{b \in A | \varphi^a(b) \neq 1\}$
6: $\phi_1 \leftarrow \phi \wedge (\varphi^a | \varphi^a)$
7: $\varphi_1 \leftarrow \varphi \wedge \big(\bigwedge_{x\in X}(\varphi^a \circ \delta_x)|(\varphi^a \circ \delta_x)\big) \wedge \big(\bigwedge_{x\in X}\bigwedge_{b\in B}(\phi_1^b \circ \delta_x)|(\phi_1^b \circ \delta_x)\big)$
8: $\phi \leftarrow \phi_1$, $\varphi \leftarrow \varphi_1$
9: **end while**
10: **return** φ

3.3 Računanje najvećih desno i levo invarijantnih fazi kvazi-uređenja

U ovoj sekciji dajemo adaptaciju prethodnih algoritama za računanje najvećih desno i levo invarijantnih fazi kvazi-uređenja. S obzirom da fazi kvazi-uređenja ne moraju zadovoljavati uslov simetričnosti, potrebno je prilagoditi pojam stabilnosti uveden u prethodnoj sekciji. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA i $\varphi, \phi \in \mathcal{Q}(A)$ dva fazi kvazi-uređenja na A. Kažemo da je *φ desno stabilna u odnosu na ϕ* ako važi

$$\varphi \leq \bigwedge_{x\in X} (\delta_x \circ \phi a)/(\delta_x \circ \phi a), \quad \text{za svako } a \in A, \tag{3.31}$$

i dualno, kažemo da je *φ levo stabilna u odnosu na ϕ* ako važi

$$\varphi \leq \bigwedge_{x\in X} (a\phi \circ \delta_x)\backslash(a\phi \circ \delta_x), \quad \text{za svako } a \in A. \tag{3.32}$$

Naredne dve Leme predstavljaju direktno proširenje Lema 3.2 i 3.3.

Lema 3.8. *Neka je $\mathcal{A}$ FKA i $\varphi \in \mathcal{Q}(A)$. Tada je φ desno (resp. levo) stabilna na $\mathcal{A}$ akko je desno (resp. levo) stabilna u odnosu na samu sebe.*

Lema 3.9. *Neka je $\mathcal{A}$ FKA i $\varphi, \phi \in \mathcal{Q}(A)$ fazi kvazi-uređenja na A takva da je $\varphi \leq \phi$. Tada je φ desno (resp. levo) stabilna u odnosu na ϕ ako je φ desno (resp. levo) stabilna.*

Naredna Teorema pruža metod za računanje najvećeg desno invarijantnog fazi kvazi-uređenja na datom fazi automatu.

Teorema 3.10. *Neka je $\mathcal{A}$ FKA. Definišimo induktivno nizove $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ fazi relacija na A na sledeći način: Za $k = 1$, odaberimo proizvoljno $a \in A$ i stavimo da je:*

$$\phi_1 = U, \tag{3.33}$$

$$\varphi_1 = \pi^{fs} \wedge \bigwedge_{x\in X} (\delta_x \circ \phi_1 a)/(\delta_x \circ \phi_1 a), \tag{3.34}$$

Pretpostavimo da su u koraku k izračunate fazi relacije ϕ_k i φ_k. U sledećem koraku, ako je $\phi_k \neq \varphi_k$, odaberimo proizvoljno $a \in A$ tako da je $\phi_k a \neq \varphi_k a$, i izračunajmo skup $B = \{b \in A | \varphi_k a(b) \neq 0\}$. Potom izračunajmo ϕ_{k+1} i φ_{k+1} na sledeći način:

$$\phi_{k+1} = \phi_k \wedge (\varphi_k a/\varphi_k a), \tag{3.35}$$

$$\varphi_{k+1} = \varphi_k \wedge \bigwedge_{x\in X}\bigwedge_{b\in B} (\delta_x \circ \phi_{k+1} b)/(\delta_x \circ \phi_{k+1} b). \tag{3.36}$$

Inače, postavimo da je $\phi_{k+1} = \phi_k$ i $\varphi_{k+1} = \varphi_k$. Tada važi sledeće:

(a) Nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ su opadajući;

(b) $\phi_k \in \mathcal{Q}(A)$ i $\varphi_k \in \mathcal{Q}(A)$, za svako $k \in \mathbb{N}$;

(c) φ_k je usitnjenje od ϕ_k, za svako $k \in \mathbb{N}$;

(d) φ_k je desno stabilna u odnosu na ϕ_k, za svako $k \in \mathbb{N}$;

(e) Za svako $k \in \mathbb{N}$ važi:

$$\phi_k \circ \delta_x \circ \varphi_k \leq \delta_x \circ \varphi_k; \tag{3.37}$$

(f) Ako je $\phi_s = \varphi_s$, za neko $s \in \mathbb{N}$, tada je $\phi_s = \varphi_s$ najveće desno invarijantno fazi kvazi-uređenje na A;

(g) Ako $\mathcal{L}(\delta, \tau)$ zadovoljava UOL, tada se nizovi $\{\phi_k\}_{k\in\mathbb{N}}$ i $\{\varphi_k\}_{k\in\mathbb{N}}$ stabilizuju, odnosno postoji $s \in \mathbb{N}$ tako da je $\phi_s = \varphi_s$.

Dokaz. (a), (b), (c), (f) i (f) dokazuju se identično kao u Teoremi 3.4.

(d) Dokazujemo da važi

$$\varphi_k \leq \bigwedge_{x\in X}\bigwedge_{a\in A} (\delta_x \circ \phi_k a)/(\delta_x \circ \phi_k a), \tag{3.38}$$

za svako $k \in \mathbb{N}$ indukcijom po k. Najpre, primetimo da je $\phi_1 = U$, te su svi aftersetovi fazi relacije ϕ_1 jednaki fazi podskupu od A čiji su svi elementi jednaki 1. Tada direktno iz definicije (3.34) sledi da za svako $a \in A$ važi:

$$\varphi_1 \leq \bigwedge_{x\in X} (\delta_x \circ \phi_1 a)/(\delta_x \circ \phi_1 a),$$

odakle (3.38) direktno sledi. Pretpostavimo sada da (3.38) važi za neko $k = m$, te dokažimo da važi i za $k = m + 1$. Najpre, primetimo da direktno iz definicije (3.36) sledi:

$$\varphi_{m+1} \leq \bigwedge_{x\in X} (\delta_x \circ \phi_{m+1} c)/(\delta_x \circ \phi_{m+1} c), \quad \text{za svako } c \in B, \tag{3.39}$$

gde su, prema konstrukciji, $a \in A$ takvo da je $\varphi_m a \neq \phi_m a$ i $B = \{b \in A | \varphi_k a(b) \neq 0\}$. Dokažimo sada da je $\phi_{m+1} c = \phi_m c$ za svako $c \in A - B$. Zaista, za svako $d \in A$ imamo da važi:

$$\begin{aligned}\phi_{m+1} c(d) &= \phi_{m+1}(d, c) = \phi_m(d, c) \wedge (\varphi_m a/\varphi_m a)(d, c)\\ &= \phi_m c(d) \wedge (\varphi_m a(c) \to \varphi_m a(d)) = \phi_m c(d) \wedge (0 \to \varphi_m a(d))\end{aligned}$$

$$= \phi_m c(d) \wedge 1 = \phi_m c(d).$$

Prema indukcijskoj pretpostavci imamo da važi

$$\varphi_m \leq \bigwedge_{x \in X} (\delta_x \circ \phi_m c) / (\delta_x \circ \phi_m c),$$

za svako $c \in A$, te i za svako $c \in A - B$. S obzirom da je $\varphi_{m+1} \leq \varphi_m$, sledi

$$\varphi_{m+1} \leq \bigwedge_{x \in X} (\delta_x \circ \phi_{m+1} c) / (\delta_x \circ \phi_{m+1} c), \quad \text{za svako } c \in A - B, \tag{3.40}$$

te na osnovu (3.39) i (3.40) dobijamo da nejednakost (3.38) važi za svako $k = m + 1$, što je i trebalo dokazati.

(e) Sledi direktno iz (d). □

Teorema 3.10 može se transformisati u proceduru za računanje najvećeg desno invarijantnog fazi kvazi-uređenja na datom FKA-u, i data je Algoritmom 3.

Algoritam 3 Računanje najvećeg desno invarijantnog fazi kvazi-uređenja na fazi automatu

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad alfabetom X i kompletnom reziduiranom mrežom $\mathcal{L}$.

Izlaz: Najveće desno invarijantno fazi kvazi-uređenje φ^{ri} na $\mathcal{A}$.

1: $\phi \leftarrow U$
2: $\varphi \leftarrow \pi^{fs} \wedge \left(\bigwedge_{x \in X} (\delta_x \circ \phi a) / (\delta_x \circ \phi a)\right)$, za proizvoljno $a \in A$
3: **while** $\phi \neq \varphi$ **do**
4: Odaberimo proizvoljno $a \in A$ tako da je $\phi a \neq \varphi a$
5: Izračunati $B = \{b \in A | \varphi a(b) \neq 0\}$
6: $\phi_1 \leftarrow \phi \wedge (\varphi a / \varphi a)$
7: $\varphi_1 \leftarrow \varphi \wedge \left(\bigwedge_{x \in X} \bigwedge_{b \in B} (\delta_x \circ \phi_1 b) / (\delta_x \circ \phi_1 b)\right)$
8: $\phi \leftarrow \phi_1$, $\varphi \leftarrow \varphi_1$
9: **end while**
10: **return** φ

Vremenska složenost Algoritma 3 može biti analizirana analogno kao za Algoritam 1, i ona je jednaka $O(lmn^5(c_\otimes + c_\vee + c_\to + c_\wedge))$, pri čemu je n broj stanja fazi automata, m broj simbola alfabeta X, a l prirodan broj takav da je broj različitih elemenata u svakom od nizova $\{\phi_k\}_{k \in \mathbb{N}}$ i $\{\varphi_k\}_{k \in \mathbb{N}}$ generisanih u Algoritmu 1 manji ili jednak od l. Takođe, važi i naredna Teorema.

Teorema 3.11. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA nad $\mathcal{L}$ i X, pri čemu je $\mathcal{L}$ BL-algebra nad realnim jediničnim intervalom $[0, 1]$. Tada su nizovi $\{\phi_k\}_{k \in \mathbb{N}}$ i $\{\varphi_k\}_{k \in \mathbb{N}}$, definisani preko* (3.33) - (3.36)*, konvergentni. Ukoliko je $\lim_{n \to \infty} \phi_n = \lim_{n \to \infty} \varphi_n = \varphi$, tada je φ najveće desno invarijantno fazi kvazi-uređenje na $\mathcal{A}$.*

Dokaz. Tvrđenje sledi direktno iz Teoreme 3.6, s obzirom da fazi kvazi-uređenja ne moraju zadovoljavati svojstvo simetričnosti. □

Primer 3.12. Neka je $\mathcal{A}$ fazi automat iz Primera 3.7. U prvom koraku, Algoritam 1 daje sledeće matrice (u matricama ϕ_k označavamo aftersetove koji se koriste u konstrukciji odgovarajućeg φ_k):

$$\phi_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \varphi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0.5 & 0.8 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Odaberimo element $a_4 \in A$ za koji je $\varphi_1^{a_4} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \neq \phi_1^{a_4}$. U sledećem koraku dobijaju se matrice:

$$\phi_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \varphi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0.5 & 0.8 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Odaberimo sada element $a_3 \in A$ jer je $\varphi_2^{a_3} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}^{-1} \neq \phi_2^{a_3}$. Dalje se dobijaju matrice:

$$\phi_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \varphi_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0.5 & 0.8 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Nadalje, za svako $k \geq 4$, odaberimo uvek element $a_1 \in A$. Tada imamo da je $\varphi_k^{a_1} = \begin{bmatrix} 1 & 0.625 & 0.5 * 0.8^{k-4} & 0 \end{bmatrix}^{-1} \neq \phi_k^{a_1}$, te je i:

$$\phi_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0.5 * 0.8^{k-4} & 0.8^{k-3} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \varphi_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0.5 * 0.8^{k-3} & 0.8^{k-2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Dakle, Algoritam 3 se ne završava u konačnom broju koraka, ali primenom Teoreme 3.11 dobijamo da je fazi relacija φ data sa:

$$\varphi = \lim_{k\to\infty} \phi_k = \lim_{k\to\infty} \varphi_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.625 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

najveće desno invarijantno fazi kvazi-uređenje na $\mathcal{A}$.

Primer 3.13. Neka je $\mathcal{A}$ fazi automat iz Primera 3.7, pri čemu je sada $\mathcal{L}$ standardna Gödelova algebra. Primenom Algoritma 3 dobija se sledeći konačan niz fazi kvazi-uređenja:

$$\phi_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \varphi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0.4 & 0.4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\phi_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.5 & 1 & 1 & 1 \\ 0.4 & 0.4 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \varphi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0.4 & 0.4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\phi_3 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0.5 & 1 & 1 & 1 \\ 0.4 & 0.4 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \varphi_3 = \phi_4 = \varphi_4 = \varphi_2 = \varphi^{\text{rifqo}},$$

te je φ^{rifqo} najveće desno invarijantno fazi kvazi-uređenje. Sa druge strane, Algoritam 1 takođe se završava u konačnom broju koraka, te dobijamo sledeći niz fazi ekvivalencija:

$$\phi_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \varphi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.4 & 0 \\ 0 & 0.4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\phi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad \varphi_2 = \phi_3 = \varphi_3 = \varphi_1 = \varphi^{\text{rife}},$$

te stoga zaključujemo da je φ^{rife} najveća desno invarijantna fazi ekvivalencija na $\mathcal{A}$.

Analogno, može se dati metod za računanje najvećeg levo invarijantnog fazi kvazi-uređenja na fazi automatu (videti Algoritam 4). Takođe, rezultat analogan Teoremi 3.11 važi i za nizove generisanim Algoritmom 4.

Algoritam 4 Računanje najvećeg levo invarijantnog fazi kvazi-uređenja na fazi automatu

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad alfabetom X i kompletnom reziduiranom mrežom $\mathcal{L}$.

Izlaz: Najveće levo invarijantno fazi kvazi-uređenje φ^{li} na $\mathcal{A}$.

1: $\phi \leftarrow U$
2: $\varphi \leftarrow \pi^{is} \wedge \big(\bigwedge_{x \in X} (a\phi \circ \delta_x) \backslash (a\varphi \circ \delta_x) \big)$, za proizvoljno $a \in A$
3: **while** $\phi \neq \varphi$ **do**
4: Odaberimo proizvoljno $a \in A$ tako da je $a\phi \neq a\varphi$
5: Izračunati $B = \{b \in A | a\varphi(b) \neq 0\}$
6: $\phi_1 \leftarrow \phi \wedge (a\varphi \backslash a\varphi)$
7: $\varphi_1 \leftarrow \varphi \wedge \big(\bigwedge_{x \in X} \bigwedge_{b \in B} (b\phi_1 \circ \delta_x) \backslash (b\phi_1 \circ \delta_x) \big)$
8: $\phi \leftarrow \phi_1$, $\varphi \leftarrow \varphi_1$
9: **end while**
10: **return** φ

3.4 Računanje najvećih desno i levo invarijantnih krisp ekvivalencija

Neka je $\mathcal{L}$ kompletna reziduirana mreža i neka je $2 = \{0,1\} \subseteq L$. Za data dva fazi podskupa $\alpha, \beta \in L^A$ nepraznog skupa A definiše se *krisp birezidual* od α i β kao krisp relacija $\alpha \uparrow \beta \in 2^{A\times A}$ sa:

$$(\alpha \uparrow \beta)(a,b) = \begin{cases} 1, & \alpha(a) = \beta(b), \\ 0, & \text{inače}, \end{cases} \tag{3.41}$$

za svako $a, b \in A$.

Teorema 3.14. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA nad $\mathcal{L}$ i X. Definišimo nizove krisp relacija $\{F_k\}_{k\in\mathbb{N}}$ i $\{E_k\}_{k\in\mathbb{N}}$, pri čemu je $F_k \in 2^{A\times A}$ i $E_k \in 2^{A\times A}$ za svako $k \in \mathbb{N}$, na sledeći način: Inicijalno za $k = 1$ stavimo da je:*

$$F_1 = U, \tag{3.42}$$

$$E_1 = (\tau \uparrow \tau) \wedge \bigwedge_{x\in X} (\delta_x \circ F_1^a) \uparrow (\delta_x \circ F_1^a), \quad \text{za neko } a \in A. \tag{3.43}$$

Dalje, za svako $k \in \mathbb{N}$ ponovimo sledeći postupak: Odaberimo $a \in A$ tako da je $F_k^a \neq E_k^a$, te postavimo da je:

$$F_{k+1} = F_k \wedge (E_k^a \uparrow E_k^a), \tag{3.44}$$

$$E_{k+1} = E_k \wedge \bigwedge_{x\in X} ((\delta_x \circ E_k^a) \uparrow (\delta_x \circ E_k^a) \wedge (\delta_x \circ (F_k^a - E_k^a)) \uparrow (\delta_x \circ (F_k^a - E_k^a))). \tag{3.45}$$

Inače, ako je $F_k = E_k$, tada stavimo da je $F_{k+1} = F_k$ i $E_{k+1} = E_k$. Tada važe sledeća tvrđenja:

(a) Nizovi $\{F_k\}_{k\in\mathbb{N}}$ i $\{E_k\}_{k\in\mathbb{N}}$ su opadajući;

(b) F_k i E_k su relacije ekvivalencije, za svako $k \in \mathbb{N}$;

(c) E_k je usitnjenje od F_k, za svako $k \in \mathbb{N}$;

(d) E_k je desno stabilna u odnosu na F_k, za svako $k \in \mathbb{N}$;

(e) Postoji $s \in \mathbb{N}$ tako da je $F_s = E_s = E$, i E je najveća desno invarijantna ekvivalencija na A.

Dokaz. (a), (b) i (c) dokazuje se analogno kao u Teoremi 3.4.

(d) Dokazujemo indukcijom po $k \in N$ da je za svako $c \in A$:

$$E_k \leq \bigwedge_{x\in X} (\delta_x \circ F_k^c) \uparrow (\delta_x \circ F_k^c); \tag{3.46}$$

Slučaj $k = 1$ sledi direktno iz definicije E_1. Za $k = 2$ imamo da je $F_2 = F_1 \wedge (E_1^a \uparrow E_1^a)$ za neko $a \in A$. Primetimo da $F_1 = U$ ima samo jednu klasu ekvivalencije, koju označavamo sa F_1^a. Prema definiciji F_2, zaključujemo da je F_2 ekvivalencija koja je dobijena deljenjem klase ekvivalencije F_1^a na dva disjunktna podskupa E_1^a i $F_1^a - E_1^a$, koji predstavljaju klase ekvivalencije od F_2. Sada, prema definiciji E_2 imamo da je:

$$E_2 = E_1 \wedge \bigwedge_{x\in X} ((\delta_x \circ (F_1^a - E_1^a)) \uparrow (\delta_x \circ (F_1^a - E_1^a))) \wedge ((\delta_x \circ E_1^a) \uparrow (\delta_x \circ E_1^a))$$

što znači da je:

$$E_2 \leq (\delta_x \circ E_1^a) \uparrow (\delta_x \circ E_1^a), \quad x \in X \tag{3.47}$$

i

$$E_2 \leq (\delta_x \circ (F_1^a - E_1^a)) \uparrow (\delta_x \circ (F_1^a - E_1^a)), \quad x \in X. \tag{3.48}$$

Iz (3.47) i (3.48) imamo da je:

$$E_2 = E_2 \wedge \bigwedge_{x \in X} ((\delta_x \circ (F_1^a - E_1^a)) \uparrow (\delta_x \circ (F_1^a - E_1^a))) \wedge ((\delta_x \circ E_1^a) \uparrow (\delta_x \circ E_1^a)),$$

što znači da za $k = 2$ važi nejednakost (3.46). Pretpostavimo da je (3.46) zadovoljena za neko $k = m$. Posmatrajmo ekvivalenciju $F_{m+1} = F_m \wedge (E_m^a | E_m^a)$ za neko $a \in A$ tako da je $F_m^a \neq E_m^a$. Prema (c) imamo da je $E_m^a < F_m^a$. To znači da je relacija ekvivalencije F_{m+1} dobijena od relacije ekvivalencije F_m deljenjem klase F_m^a na dve klase E_m^a i $F_m^a - E_m^a$. Stoga, za svaku klasu ekvivalencije F_{m+1}^b, $b \in A$, za koju je $F_{m+1}^b \neq E_m^a$ i $F_{m+1}^b \neq (F_m^a - E_m^a)$, prema indukcijskoj hipotezi imamo da važi:

$$\begin{aligned} E_{m+1} \leq E_m &= E_m \wedge \bigwedge_{x \in X} (\delta_x \circ F_m^b) \uparrow (\delta_x \circ F_m^b) \\ &= E_m \wedge \bigwedge_{x \in X} (\delta_x \circ F_{m+1}^b) \uparrow (\delta_x \circ F_{m+1}^b) \leq \bigwedge_{x \in X} (\delta_x \circ F_{m+1}^b) \uparrow (\delta_x \circ F_{m+1}^b), \end{aligned}$$

te je stoga:

$$E_{m+1} = E_{m+1} \wedge \bigwedge_{x \in X} (\delta_x \circ F_{m+1}^b) \uparrow (\delta_x \circ F_{m+1}^b).$$

Dakle, za svako $b \in A$ za koje je $R_{m+1}^b \neq E_m^a$ i $F_{m+1}^b \neq (F_m^a - E_m^a)$, (3.46) važi. Što se klase ekvivalencije E_m^a tiče, prema definiciji E_{m+1} važi:

$$E_{m+1} \leq E_m \wedge \bigwedge_{x \in X} (\delta_x \circ E_m^a) \uparrow (\delta_x \circ E_m^a) \leq \bigwedge_{x \in X} (\delta_x \circ E_m^a) \uparrow (\delta_x \circ E_m^a),$$

što znači da je:

$$E_{m+1} = E_{m+1} \wedge \bigwedge_{x \in X} (\delta_x \circ E_m^a) \uparrow (\delta_x \circ E_m^a).$$

Na analogan način dokazujemo i da je:

$$E_{m+1} = E_{m+1} \wedge \bigwedge_{x \in X} (\delta_x \circ (F_m^a - E_m^a)) \uparrow (\delta_x \circ (F_m^a - E_m^a)).$$

Dakle, za $k = m + 1$ takođe važi nejednakost (3.46), što je i trebalo dokazati.

(e) Prema (a), niz $\{E_k\}_{k \in N}$ je opadajući. S obzirom da ekvivalencija E_1 ima u najgorem slučaju jednu klasu, sledi da se niz $\{E_k\}_{k \in N}$ stabilizuje najviše nakon $|A| - 1$ koraka, jer je $|A|$ ukupan broj klasa ekvivalencije od E_k. Preostali deo tvrđenja dokazuje se analogno kao u Teoremi 3.4(e). □

Teorema 3.14 može se transformisati u naredni algoritam za računanje najveće desno invarijantne ekvivalencije na fazi automatu.

Odredimo sada vremensku složenost Algortima 5. Najpre primetimo da se u analizi računanja kompozicije $\delta_x \circ F^a$ mogu izostaviti $c_\otimes$, $c_\rightarrow$ i $c_\wedge$, s obzirom da je $F^a \in 2^A$ te se množenje vrši jedino sa 0 i 1. Dakle, korak 2 izvršava se u $O(mn^2 c_\vee))$ vremenu. Slično, korak 5 izvršava se u $O(n^2)$ vremenu, dok se korak 6 izvršava u $O(mn^2 c_\vee)$ vremenu. Kao što je pokazano u Teoremi 3.14(e), ukupan broj puta kroz koji prolazi petlja kroz korake 3-7 Algoritma 5 je najviše $n - 1$, tj. vreme izvršenja tog dela jednako je $O(n)$. To znači da je vreme izvršenja celog algoritma $O(mn^3 c_\vee)$.

Algoritam 5 Računanje najveće desno invarijantne ekvivalencije na fazi automatu

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad alfabetom X i kompletnom reziduiranom mrežom $\mathcal{L}$.

Izlaz: Najveća desno invarijantna ekvivalencija E^{ri} na $\mathcal{A}$.

1: $F \leftarrow U$
2: $E \leftarrow (\tau \uparrow \tau) \wedge \big(\bigwedge_{x \in X}(\delta_x \circ F^a) \uparrow (\delta_x \circ F^a)\big)$, za proizvoljno $a \in A$
3: **while** $F \neq E$ **do**
4: Odaberimo proizvoljno $a \in A$ tako da je $F^a \neq E^a$
5: $F_1 \leftarrow F \wedge (E^a \uparrow E^a)$
6: $E_1 \leftarrow E \wedge \bigwedge_{x \in X} \big((\delta_x \circ E^a) \uparrow (\delta_x \circ E^a) \wedge (\delta_x \circ (F^a - E^a)) \uparrow (\delta_x \circ (F^a - E^a))$
7: $F \leftarrow F_1$, $E \leftarrow E_1$
8: **end while**
9: **return** E

S obzirom da su u radu [45] autori dali algoritam za računanje najveće desno invarijantne ekvivalencije na fazi automatu koji se izvršava u $O(mn^5 c_\vee)$ vremenu, zaključujemo da dati algoritam predstavlja poboljšanje već postojećeg algoritma iz [45].

Algoritam 5 pokazuje da se najveća desno invarijantna ekvivalencija može efikasno izračunati za bilo koji fazi automat, i da se završava u konačnom broju koraka bez nametanja dodatnih uslova kompletnoj reziduiranoj mreži $\mathcal{L}$. Međutim, s obzirom da važi $E^{\mathrm{ri}} \leq \varphi^{\mathrm{ri}}$, na osnovu Teoreme 4.13, zaključujemo da desno invarijantne ekvivalencije ne moraju biti efikasne kao desno invarijantne fazi ekvivalencije u determinizaciji fazi automata.

Glava 4

Determinizacioni algoritmi za fazi automate

Za razliku od NKA-a, koji uvek mogu biti determinizovani, ne mogu svi FKA-i biti determinizovani. Stoga je problem determinizacije od posebnog značaja u kontekstu FKA-a, te su tokom godina razvijeni brojni metodi za determinizaciju FKA-a.

Determinizacija FKA-a primarno je posmatrana kao njegova konverzija u ekvivalentan *krisp-deterministički fazi konačni automat* (k-DFKA, skraćeno) od strane brojnih autora, i ovakvi algoritmi su dobro razvijeni u skorašnjoj literaturi. Preciznije, Malik i Mordenson [140, 157] pokazali su da je svaki $(\max, \min)$-fazi automat ekvivalentan k-DFKA-u, dok su istu tvrdnju Bozapalidis i Louscou-Bozapalidou pokazali za $(\max, T)$- i $(\max, D)$-fazi automate, pri čemu T označava Łukasiewiczevu normu, dok D označava drastični presek. Bělohlávek [14] je posmatrao fazi automate nad lokalno konačnim kompletnim mrežama i pokazao njihovu ekvivalentnost sa k-DFKA, dok je isti rezultat dao Rahonis [177] za fazi automate nad ograničenim distributivnim mrežama. Li i Pedrycz [135] posmatrali su fazi automate nad mrežno uređenim monoidima i pokazali da je takav fazi automat ekvivalentan sa k-DFKA ako je poluprstenski ostatak monoida lokalno konačan. Metodi za determinizaciju FKA koji su razvili Bělohlávek [14] i Li i Pedrycz [135] analogni su dobro poznatoj *podskupovnoj konstrukciji* za NKA-e.

Ignjatović, Ćirić i Bogdanović razvili su u [97] determinizacioni algoritam za FKA-e nad kompetnim reziduiranim mrežama, a koji je analogan *dostižnoj podskupovnoj konstrukciji* za NKA-e, i pokazali da je determinizacija moguća u slučaju kada Nerodova fazi desna kongruencija na FKA ima konačan indeks (videti i [100]). Metod dat u radu [97] poboljšan je u brojnim radovima. Jančić, Ignjatović i Ćirić [109] razvili su algoritam za konstrukciju *dečjeg fazi automata* koji generalizuje determinizaciju preko skupova prelaza [75, 74]. Potom su Jančić i ostali u [110] razvili algoritme za konstrukciju k-DFA-a koristeći desno i levo invarijantne fazi relacije, a koji generalizuju determinizaciju preko simulacija [75, 74]. Algoritmi razvijeni u [110] zapravo kombinuju determinizacioni metod i metod redukcije broja stanja fazi automata u jedan metod koji simultano vrši ove dve operacije, čime se zapravo dobijaju algoritmi koji su bolji od algoritama razvijenih u [14, 97, 109, 135], u smislu da generišu determinističke fazi automate sa manjim brojem stanja dok rade u istoj vremenskoj složenosti.

Prethodni algoritmi ne mogu se koristiti ukoliko ulazni FKA prihvata fazi jezik beskonačnog ranga, s obzirom da k-DFKA po svojoj prirodi raspoznaje jedino fazi jezik konačnog ranga. Stoga, de Mendívil i Garitagoitia razvijaju u [145] determinizacioni metod koji konvertuje ulazni FKA u ekvivalentan KDFA, s obzirom da je KDFA u stanju da raspozna fazi jezik beskonačnog ranga. Njihov algoritam bazira se na pojmu *faktorizacije fazi podskupova*, koji su prvobitno uveli Kirsten i Mäurer [116] u determinizacionom metodu za težinske automate nad poluprstenima. Osim toga,

de Mendívil i Garitagoitia [145] pokazuju i da je *svojstvo reprezentativnih ciklusa* potreban i dovoljan uslov da se njihov determinizacioni metod završi u konačnom broju koraka. Ovaj uslov opštiji je od *svojstva blizanaca*, a koji predstavlja dovoljan uslov za determinizaciju težinskih automata nad polupstenima. Preciznije, autori su u [145] dali primere fazi automata koji ne zadovoljavaju svojstvo blizanaca a zadovoljavaju svojstvo reprezentativnih ciklusa, i za koje se determinizacioni metod završava u konačnom broju koraka.

U ovoj glavi dajemo algoritme za determinizaciju FKA koji raspoznaju fazi jezike beskonačnog ranga. Drugim rečima, pod determinizacijom podrazumevamo konverziju FKA u KDFA. Prikazani algoritmi zasnivaju se na korišćenju faktorizacije fazi podskupova, kao i desno invarijantnih fazi kvazi-uređenja uvedenih u Glavi 3, i zapravo vrše simultano determinizaciju i sažimanje ekvivalentnih stanja fazi automata u konstrukciji. Na ovaj način, poboljšava se determinizacioni algoritam razvijen u [145]. Osim toga, uvodimo i uslov *slabo svojstvo reprezentativnih ciklusa*, te dokazujemo da je ovaj uslov potreban i dovoljan da ulazni FKA ima ekvivalentan KDFKA. Ovaj uslov je još opštiji od svojstva reprezentativnih ciklusa, čime se proširuje klasa fazi automata koji mogu biti determinizovani.

Rezultati iz ove glave objavljeni su u radu [198], a organizovani su na sledeći način. U Sekciji 4.1 daju se definicije faktorizacije i maksimalne faktorizacije fazi podskupova nad kompletnim reziduiranim mrežama bez delioca nule, kao i njihova osnovna svojstva i osobine. U Sekciji 4.2 prikazan je način za determinizaciju fazi automata na osnovu faktorizacije fazi podskupova, kao i desno invarijantnih fazi kvazi-uređenja. Preciznije, za dati FKA $\mathcal{A}$ čiji je skup stanja A, kao i fazi relaciju φ na A, najpre konstruišemo FKA $\mathcal{A}_\varphi$, a potom konstruišemo kompletan deterministički fazi automat $\mathcal{A}_\varphi^D$ od $\mathcal{A}$. Takođe, u Sekciji 4.3, za dati FKA $\mathcal{A}$ i $\varphi \in L^{A\times A}$, konstruišemo takozvani kompletan deterministički dečji fazi automat $(\mathcal{A}_\varphi^D)^c$ od $\mathcal{A}$, za datu faktorizaciju D. Dokazujemo da su oba fazi automata $\mathcal{A}_\varphi^D$ i $(\mathcal{A}_\varphi^D)^c$ ekvivalentna sa $\mathcal{A}$ u slučaju kada je φ refleksivna slabo desno invarijantna fazi relacija na A. Takođe, dokazujemo da je $(\mathcal{A}_\varphi^D)^c$ konačan ako i samo ako je $\mathcal{A}_\varphi^D$ konačan, kao i da broj stanja od $(\mathcal{A}_\varphi^D)^c$ ne može biti veći od broja stanja od $\mathcal{A}_\varphi^D$. Štaviše, dokazujemo da korišćenjem najvećih desno invarijantna fazi kvazi-uređenja algoritmi rezultiraju najmanjim KDFA-om, čime se direktno daje mogućnost primene algoritama razvijenih u Glavi 3. Sekcija 4.4 posvećena je formalnoj analizi razvijenih algoritama. Na kraju, u Sekciji 4.5 uvodimo takozvano *slabo svojstvo reprezentativnih ciklusa*, i dokazujemo da su $\mathcal{A}_\varphi^D$ i $(\mathcal{A}_\varphi^D)^c$ konačni, pod uslovom da je D maksimalna faktorizacija, ako i samo ako $\mathcal{A}_\varphi$ zadovoljava slabo svojstvo reprezentativnih ciklusa. Ovo svojstvo opštije je od svojstva reprezentativnih ciklusa prethodno određenog u [144] kao potreban i dovoljan uslov za determinizaciju preko maksimalne faktorizacije, čime proširujemo klasu fazi automata koji mogu biti determinizovani. Ovo svojstvo formuliše se jedino na osnovu interne strukture ulaznog fazi automata. Na kraju, u Sekciji 4.6 prikazani su brojni ilustrativni primeri.

4.1 Faktorizacije fazi podskupova

U nastavku teksta, ukoliko nije drugačije naglašeno, posmatramo kompletne reziduirane mreže bez delioca nule.

Definicija 4.1. Neka je $\mathcal{L}$ kompletna reziduirana mreža i A neprazan skup. Tada je *faktorizacija fazi podskupova od A nad $\mathcal{L}$*, ili samo *faktorizacija od A*, uređen par $D =$

(f, g) preslikavanja $f : L^A \to L^A$ i $g : L^A \to L$ takav da važi:

$$\alpha = g(\alpha) \otimes f(\alpha), \quad \text{za svako } \alpha \in L^A, \tag{4.1}$$
$$g(\emptyset) = 1. \tag{4.2}$$

Sledeća Lema navodi osnovna svojstva faktorizacije nad kompletnim reziduiranim mrežama.

Lema 4.2. *Neka je $D = (f, g)$ faktorizacija od A, pri čemu je A neprazan skup. Tada za svako $\alpha \in L^A$ važi:*

$$g(\alpha) > 0, \tag{4.3}$$
$$\alpha(a) = 0 \quad \textit{akko} \quad f(\alpha)(a) = 0, \quad \textit{za svako } a \in A, \tag{4.4}$$
$$\alpha = \emptyset \quad \textit{akko} \quad f(\alpha) = \emptyset, \tag{4.5}$$
$$\alpha \leq f(\alpha), \tag{4.6}$$
$$f(\alpha)(a) \leq g(\alpha) \to \alpha(a), \quad \textit{za svako } a \in A, \tag{4.7}$$
$$g(\alpha) \geq \bigvee_{a \in A} \alpha(a). \tag{4.8}$$

Dokaz. (4.3): Direktno iz (4.1) i (1.44) zaključujemo da važi:

$$\alpha(a) = g(\alpha) \otimes f(\alpha)(a), \quad \text{za svako } a \in A. \tag{4.9}$$

Pretpostavimo da za neki fazi podskup α od A važi $g(\alpha) = 0$. Tada prema (1.22) i (4.9) važi $\alpha = 0 \otimes f(\alpha) = \emptyset$. Ali tada, prema (4.2), važi $g(\alpha) = 1$, što dovodi do kontradikcije.

(4.4): Ako je $f(\alpha)(a) = 0$, za neko $a \in A$, tada iz (4.9) sledi $\alpha(a) = 0$. Obratno, ako važi $\alpha(a) = 0$ za neko $a \in A$, tada je takođe iz (4.9) $f(\alpha)(a) = 0$ jer iz (4.3) sledi $g(\alpha) > 0$, a $\mathcal{L}$ je bez delioca nule.

(4.5): Direktna posledica (4.4).

(4.6): Sledi iz (4.9) i (1.18).

(4.7): Sledi iz (4.9) i svojstva adjunkcije (1.11).

(4.8): Takođe iz (4.9) i (1.18) sledi $\alpha(a) \leq g(\alpha)$, za svako $a \in A$, što znači da je $g(\alpha) \in \mathcal{U}(\text{rang}(\alpha))$. Sada dokaz sledi iz činjenice da je $\bigvee_{a \in A} \alpha(a)$, po definiciji, najmanji element skupa $\mathcal{U}(\text{rang}(\alpha))$. □

U svakoj kompletnoj reziduiranoj mreži postoji *trivijalna faktorizacija*, odnosno faktorizacija $D_N = (f_N, g_N)$ od A, gde je $g_N(\alpha) = 1$ i $f_N(\alpha) = \alpha$, za svako $\alpha \in L^A$.

Proizvoljna faktorizacija $D = (f, g)$ od A naziva se *maksimalna* ako važi:

$$g(x \otimes \alpha) = x \otimes g(\alpha), \tag{4.10}$$
$$f(x \otimes \alpha) = f(\alpha), \tag{4.11}$$

za svako $x \in L$ i $\alpha \in L^A$ za koje je $x \otimes \alpha \neq \emptyset$. Uslov $x \otimes \alpha \neq \emptyset$ je neophodan, jer bi bez tog uslova imali da je $f(\alpha) = f(0 \otimes \alpha) = f(\emptyset)$, te oznaka maksimalne faktorizacije ne bi imala smisla. Maksimalne faktorizacije igraju važnu ulogu u determinizaciji fazi automata, kao što će biti formalno dokazano u narednim sekcijama.

Lema 4.3. *Neka je $D = (f, g)$ maksimalna faktorizacija od nepraznog skupa A. Tada za svako $\alpha \in L^A$ važi:*

$$f(f(\alpha)) = f(\alpha). \tag{4.12}$$

Dokaz. U slučaju kada je $\alpha = \emptyset$, prema (4.5) važi $f(f(\alpha)) = f(\emptyset) = \emptyset = f(\alpha)$. Inače, ako je $\alpha \neq \emptyset$, tada je

$$f(\alpha) = f(g(\alpha) \otimes f(\alpha)).$$

Iz (4.3) i $\alpha \neq \emptyset$ zaključujemo da je $g(\alpha) \otimes f(\alpha) \neq 0$. S obzirom da je D maksimalna faktorizacija od A, (4.11) takođe važi, sledi da (4.12) važi. □

Posmatrajmo sada problem konstrukcije faktorizacije nad kompletnom reziduiranom mrežom bez delioca nule. Definišimo funkcije $g_M : L^A \to L$ i $f_M : L^A \to L^A$ sa

$$g_M(\alpha) = \begin{cases} 1, & \alpha = \emptyset \\ \bigvee_{a \in A} \alpha(a), & \text{inače} \end{cases}, \qquad f_M(\alpha)(a) = g_M(\alpha) \to \alpha(a), \tag{4.13}$$

za svako $\alpha \in L^A$ i $a \in A$. Tada važi sledeće tvrđenje.

Lema 4.4. *Neka je $\mathcal{L}$ kompletna reziduirana mreža bez delioca nule i koja zadovoljava svojstvo deljivosti, i neka je A neprazan skup. Ukoliko $\mathcal{L}$ zadovoljava svojstvo deljivosti, tada je $D_M = (f_M, g_M)$, definisan sa* (4.13)*, faktorizacija od A.*

Dokaz. Na osnovu definicije (4.13) zaključujemo da važi $\alpha(a) \leq g_M(\alpha)$, za svako $a \in A$ i $\alpha \in L^A$. S obzirom da je svojstvo deljivosti (1.38) zadovoljeno, sledi da postoji $z \in L$ tako da je $\alpha(a) = g_M(\alpha) \otimes z$. Tada iz (1.24) sledi:

$$g_M(\alpha) \otimes f_M(\alpha)(a) = g_M(\alpha) \otimes (g_M(\alpha) \to \alpha(a)) = \alpha(a),$$

za svako $a \in A$ i $\alpha \in L^A$. Dakle, svojstvo (4.1) je zadovoljeno. Svojstvo (4.2) je već po definiciji g_M zadovoljeno. Sledi, $D_M = (f_M, g_M)$ je faktorizacija od A. □

Prema tome, uređeni par $D_M = (f_M, g_M)$ nazivamo *Mohrieva faktorizacija* kada je $\mathcal{L}$ kompletna reziduirana mreža bez delioca nule i koja zadovoljava svojstvo deljivosti. U opštem slučaju, D_M nije maksimalna faktorizacija, ali su zadovoljena sledeća svojstva.

Lema 4.5. *Za svako $\alpha \in L^A$ i $x \in L$, Mohrieva faktorizacija $D_M = (f_M, g_M)$ zadovoljava:*

$$g_M(x \otimes \alpha) = x \otimes g_M(\alpha), \tag{4.14}$$
$$f_M(\alpha) \leq f_M(x \otimes \alpha), \tag{4.15}$$
$$f_M(\alpha) = f_M(f_M(\alpha)). \tag{4.16}$$

Dokaz. Na osnovu (1.30) sledi da za svako $x \in L$ i $\alpha \in L^A$ za koje je $x \otimes \alpha \neq \emptyset$ važi:

$$g_M(x \otimes \alpha) = \bigvee_{a \in A} (x \otimes \alpha)(a) = \bigvee_{a \in A} x \otimes \alpha(a) = x \otimes \bigvee_{a \in A} \alpha(a) = x \otimes g_M(\alpha),$$

svojstvo (4.14) sledi. Dalje, za svako $a \in A$, primenjujući redom (1.20), (1.23) i (4.14), dobijamo:

$$\begin{aligned} f_M(\alpha)(a) &= g_M(\sigma) \to \sigma(a) = 1 \otimes (g_M(\alpha) \to \alpha(a)) \\ &= (x \to x) \otimes (g_M(\alpha) \to \alpha(a)) \\ &\leq (x \otimes g_M(\alpha)) \to (x \otimes \alpha(a)) \\ &= f_M(x \otimes \alpha)(a), \end{aligned}$$

odakle (4.15) sledi. Na kraju, da bismo pokazali da (4.16) važi, dokazujemo obe strane nejednakosti. S jedne strane, s obzirom da je $g_M(f_M(\alpha)) \leq 1$, iz (1.21) i (1.29) sledi

$$f_M(\alpha)(a) = 1 \to f_M(\alpha)(a) \leq g_M(f_M(\alpha)) \to f_M(\alpha)(a) = f_M(f_M(\alpha))(a),$$

za svako $a \in A$. Sa druge strane, iz (4.15) i činjenica da je $D_M = (f_M, g_M)$ faktorizacija od A, sledi

$$f_M(f_M(\alpha)) \leq f_M(g_M(\alpha) \otimes f_M(\alpha)) = f_M(\alpha),$$

čime je dokaz završen. □

Propozicija 4.6. *Neka je $\mathcal{L}$ standardna prozvod algebra, definisana sa* (1.14)*, i A neprazan skup. Tada je Mohrijeva faktorizacija $D_M = (f_M, g_M)$ maksimalna.*

Dokaz. S obzirom da je $\alpha(a) \leq g_M(\alpha)$, za svako $a \in A$ i $\alpha \in L^A$, iz (1.14) sledi da je

$$f_M(\alpha)(a) = \frac{\alpha(a)}{g_M(\alpha)},$$

te za svako $x \in L$, $a \in A$ i $\alpha \in L^A$ važi:

$$f_M(x \otimes \alpha)(a) = \frac{(x \otimes \alpha)(a)}{g_M(x \otimes \alpha)} = \frac{x \cdot \alpha(a)}{x \cdot g_M(\alpha)} = \frac{\alpha(a)}{g_M(\alpha)} = f_M(\alpha)(a),$$

dakle, D_M je maksimalna faktorizacija od A. □

4.2 Determinizacija preko faktorizacije fazi podskupova i desno invarijantnih fazi kvazi-uređenja

Neka je $\mathcal{L}$ kompletna reziduirana mreža bez delioca nule i koja zadovoljava svojstvo deljivosti, i neka je X konačan alfabet. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat nad $\mathcal{L}$ i X, i neka je $D = (f, g)$ faktorizacija od A. Definišimo familiju $\{\sigma_u^D\}_{u \in X^*}$ fazi podskupova od A induktivno sa:

$$\begin{aligned} &\sigma_\varepsilon^D = f(\sigma), \\ &\sigma_{ux}^D = f(\sigma_u^D \circ \delta_x), \quad \text{za svako } u \in X^* \text{ i } x \in X. \end{aligned}$$

Drugim rečima, za svako $u = x_1 x_2 \dots x_n$, pri čemu je $x_1, x_2, \dots, x_n \in X$ i $n \in \mathbb{N}_0$, imamo da važi:

$$\sigma_u^D = f(\dots f(f(f(\sigma) \circ \delta_{x_1}) \circ \delta_{x_2}) \circ \dots) \circ \delta_{x_n}).$$

Označimo sa $A^D = \{\sigma_u^D\}_{u \in X^*}$, i definišimo funkciju fazi prelaza $\delta^D : A^D \times X \times A^D \to L$ sa

$$\delta^D(\alpha, x, \beta) = \begin{cases} g(\sigma_u^D \circ \delta_x), & \alpha = \sigma_u^D \text{ and } \beta = \sigma_{ux}^D, \\ 0, & \text{inače} \end{cases} \tag{4.17}$$

za svako $u \in X^*$ i $x \in X$, kao i fazi skup završnih stanja $\tau^D : A^D \to L$ sa:

$$\tau^D(\alpha) = \alpha \circ \tau,$$

za svako $\alpha \in A^D$. Tada važi sledeći rezultat.

Teorema 4.7. *Neka je $\mathcal{A}$ fazi automat i $D = (f, g)$ faktorizacija od A. Tada je $\mathcal{A}^D = (A^D, \{\sigma_\varepsilon^D / g(\sigma)\}, \delta^D, \tau^D)$ dobro definisan KDFA.*

Dokaz. (Dobro definisan) Očigledno je da su preslikavanja δ^D i τ^D dobro definisana, te je fazi automat $\mathcal{A}^D$ dobro definisan fazi automat.
(Kompletan) Za svako $u \in X^*$ i $x \in X$ imamo da je $\sigma_u^D \in A^D$ i $\sigma_{ux}^D \in A^D$, tada iz (4.3) i (4.17) sledi $\delta^D(\sigma_u^D, x, \sigma_{ux}^D) = g(\sigma_u^D \circ \delta_x) > 0$.
(Deterministički) Prema konstrukciji fazi automata $\mathcal{A}^D$, jedinstveno početno stanje je σ_ε^D sa stepenom $g(\sigma)$, a prema (4.3), taj stepen je veći od 0. Staviše, ako je $\delta^D(\sigma_u^D, x, \sigma_1) > 0$ i $\delta^D(\sigma_u^D, x, \sigma_2) > 0$, za neko $u \in X^*$, $x \in X$ i $\sigma_1, \sigma_2 \in A^D$, tada prema (4.17) imamo $\sigma_1 = \sigma_2 = \sigma_{ux}^D$. □

Determinizacioni metod baziran na gore opisanoj konstrukciji nazvan je u [144] *determinizacija preko faktorizacije fazi stanja*. U slučaju kada je D trivijalna faktorizacija, fazi automat $\mathcal{A}^D$ postaje *Nerodov fazi automat* $\mathcal{A}^N$ konstruisan u [97] (videti i [100]). Dakle, fazi automat $\mathcal{A}^D$ predstavlja generalizaciju Nerodovog fazi automata $\mathcal{A}^N$. Fazi automat $\mathcal{A}^D$ naziva se *KDFA od* $\mathcal{A}$. S obzirom da je $\mathcal{A}^D$ KDFA, prema (2.12) imamo da je fazi jezik raspoznat fazi automatom $\mathcal{A}^D$ jednak:

$$[\![\mathcal{A}^D]\!](u) = c_u^D \otimes \tau^D(\sigma_u^D), \tag{4.18}$$

za svaku reč $u = x_1 x_2 \dots x_n \in X^*$, pri čemu je skalar $c_u^D \in \mathcal{L}$ definisan sa:

$$c_u^D = \sigma^D(\sigma_\varepsilon^D) \otimes \left(\bigotimes_{i=1}^{n} \delta^D(\sigma_{x_1 x_2 \dots x_{i-1}}^D, x_i, \sigma_{x_1 x_2 \dots x_i}^D) \right). \tag{4.19}$$

Na osnovu prethodnih zapažanja, zaključujemo da važi $c_u^D > 0$.

Lema 4.8. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, i neka je $\mathcal{A}^D$ KDFA od $\mathcal{A}$. Tada za svako $u \in X^*$ važi:*

$$\sigma_u = \sigma \circ \bigcirc_{i=1}^{n} \delta_{x_i} = c_u^D \otimes \sigma_u^D. \tag{4.20}$$

gde je c_u^D definisan sa (4.19).

Dokaz. Dokaz se izvodi indukcijom po dužini $n \in \mathbb{N}_0$ reči $u \in X^*$. Kada je $n = 0$, počevši od leve strane jednakosti (4.20) dobijamo:

$$\sigma = g(\sigma) \otimes f(\sigma) = \sigma^D(\sigma_\varepsilon^D) \otimes \sigma_\varepsilon^D,$$

što je tačno desna strana jednakosti (4.20), dakle (4.20) važi za $n = 0$. Pretpostavimo sada da (4.20) važi za neko $n \in \mathbb{N}_0$ i simbole $x_1, x_2, \dots, x_n \in X$. Tada prema indukcijskoj hipotezi za svako $x_{n+1} \in X$ važi:

$$\sigma \circ \left(\bigcirc_{i=1}^{n+1} \delta_{x_i} \right) = \sigma \circ \left(\bigcirc_{i=1}^{n} \delta_{x_i} \right) \circ \delta_{x_{n+1}} = c_{x_1 x_2 \dots x_n}^D \otimes (\sigma_{x_1 x_2 \dots x_n}^D \circ \delta_{x_{n+1}}).$$

Štaviše, s obzirom da je $D = (f, g)$ proizvoljna faktorizacija, sledi:

$$\sigma \circ \left(\bigcirc_{i=1}^{n+1} \delta_{x_i} \right) = c_{x_1 x_2 \dots x_n}^D \otimes \left(g(\sigma_{x_1 x_2 \dots x_n}^D \circ \delta_{x_{n+1}}) \otimes f(\sigma_{x_1 x_2 \dots x_n}^D \circ \delta_{x_{n+1}}) \right).$$

Na kraju, prema asocijativnosti $\otimes$, dobijamo:

$$\sigma \circ \left(\bigcirc_{i=1}^{n+1} \delta_{x_i} \right) = \left(c^D_{x_1 x_2 \ldots x_n} \otimes \delta^D(\sigma^D_{x_1 x_2 \ldots x_n}, x_{n+1}, \sigma^D_{x_1 x_2 \ldots x_{n+1}}) \right) \otimes \sigma^D_{x_1 x_2 \ldots x_{n+1}}.$$

S obzirom da $c^D_{x_1 x_2 \ldots x_n} \otimes \delta^D(\sigma^D_{x_1 x_2 \ldots x_n}, x_{n+1}, \sigma^D_{x_1 x_2 \ldots x_{n+1}}) = c^D_{x_1 x_2 \ldots x_n x_{n+1}}$ važi prema asocijativnosti $\otimes$, dokaz je završen. □

Ekvivalencija fazi automata $\mathcal{A}$ i $\mathcal{A}^D$ sledi iz naredne Teoreme.

Teorema 4.9. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, i $\mathcal{A}^D$ KDFA od $\mathcal{A}$. Tada važi $[\![\mathcal{A}]\!] = [\![\mathcal{A}^D]\!]$.*

Dokaz. Odaberimo proizvoljnu reč $u \in X^*$. Primenom (4.20) dobija se

$$[\![\mathcal{A}]\!](u) = \sigma \circ \delta_u \circ \tau = c^D_u \otimes (\sigma^D_u \circ \tau).$$

Imajući u vidu da je $\tau^D(\sigma^D_u) = \sigma^D_u \circ \tau$, kao i da (4.18) važi, zaključujemo da je $[\![\mathcal{A}]\!](u) = [\![\mathcal{A}^D]\!](u)$. S obzirom da je reč $u \in X^*$ odabrana na proizvoljan način, ekvivalencija fazi automata $\mathcal{A}$ i $\mathcal{A}^D$ sledi. □

Maksimalne faktorizacije igraju važnu ulogu u determinizaciji fazi automata, kao što je prikazano u narednoj Teoremi. Pre toga je potrebno dokazati narednu Lemu.

Lema 4.10. *Neka je $\mathcal{A}$ fazi automat i D maksimalna faktorizacija od A. Tada za svako $u \in X^*$ važi:*

$$f(\sigma^D_u) = \sigma^D_u. \tag{4.21}$$

Dokaz. Najpre, primetimo da u slučajevima kada je $\sigma^D_u = \emptyset$, pri čemu je $u \in X^*$, iz (4.5) dobijamo $f(\sigma^D_u) = \emptyset = \sigma^D_u$. Pretpostavimo da je $\sigma^D_u \neq \emptyset$. Ako je $u = \varepsilon$, iz svojstva (4.12) sledi:

$$f(\sigma^D_u) = f(f(\sigma)) = f(\sigma) = \sigma^D_u.$$

Inače, ako je $u \in X^+$, tada se reč u može predstaviti kao $u = x_1 x_2 \ldots x_n$, pri čemu je $n > 0$, te tada ponovo iz (4.12) sledi:

$$f(\sigma^D_u) = f(f(\sigma^D_{x_1 x_2 \ldots x_{n-1}} \circ \delta_{x_n})) = f(\sigma^D_{x_1 x_2 \ldots x_{n-1}} \circ \delta_{x_n}) = \sigma^D_u,$$

čime je dokaz završen. □

Teorema 4.11. *Neka je $\mathcal{A}$ fazi automat, $N = (f_N, g_N)$, $M = (f_M, g_M)$ i $D = (f, g)$ trivijalna, maksimalna i proizvoljna faktorizacija od A, respektivno, i $\mathcal{A}^N$, $\mathcal{A}^M$ and $\mathcal{A}^D$ odgovarajući KDFA-i od $\mathcal{A}$, respektivno. Tada važi:*

$$|\mathcal{A}^M| \leq |\mathcal{A}^D| \quad i \quad |\mathcal{A}^M| \leq |\mathcal{A}^N|. \tag{4.22}$$

Dokaz. Za proizvoljnu reč $u \in X^*$, prema Lemi 4.8 dobijamo da važi:

$$\sigma_u = c^D_u \otimes \sigma^D_u = c^M_u \otimes \sigma^M_u.$$

S obzirom da je D_M maksimalna faktorizacija od A, iz (4.11) sledi

$$f_M(c^D_u \otimes \sigma^D_u) = f_M(\sigma^D_u),$$

i takođe:

$$f_M(c_u^M \otimes \sigma_u^M) = f_M(\sigma_u^M) = \sigma_u^M,$$

stoga važi:

$$\sigma_u^M = f_M(\sigma_u^D) = f_M(\sigma_u).$$

S obzirom da je prethodni identitet zadovoljen za svako $u \in X^*$, sledi da su nejednakosti (4.22) zadovoljene. □

Teorema 4.11 navodi da KDFA od $\mathcal{A}$ dobijen preko maksimalne faktorizacije nema veći broj stanja od KDFA od $\mathcal{A}$ dobijen preko proizvoljne faktorizacije ili fazi Nerodovog automata od $\mathcal{A}$ (iako KDFA od $\mathcal{A}$ dobijen preko proizvoljne faktorizacije može imate veći broj stanja od Nerodovog fazi automata od $\mathcal{A}$, kao što je pokazano primerima u Sekciji 4.6). Ova činjenica je od posebnog interesa u standardnoj proizvod algebri, koja nije lokalno konačna. Zaista, kada je $\mathcal{A}$ fazi automat definisan nad standardnom proizvod algebrom i postoji barem jedan ciklus u grafu prelaza od $\mathcal{A}$, tada je Nerodov fazi automat $\mathcal{A}^N$ beskonačan, ali $\mathcal{A}^M$ može biti konačan, s obzirom da nema veći broj stanja od $\mathcal{A}^N$. Ipak, postoje slučajevi kada je i fazi automat $\mathcal{A}^M$ beskonačan. Stoga je od posebne važnosti razvoj determinizacionih metoda koji će biti u stanju da prevaziđu ove nedostatke.

Stoga obraćamo pažnju na sledeću konstrukciju. Neka je φ proizvoljna fazi relacija na A. Definišimo fazi automat $\mathcal{A}_\varphi = (A, \sigma_\varphi, \delta_\varphi, \tau_\varphi)$ na sledeći način:

$$\begin{aligned}
&\sigma_\varphi = \sigma \circ \varphi,\\
&\delta_\varphi(a, x, b) = (\delta_x \circ \varphi)(a, b), \quad \text{za svako } a, b \in A \text{ i } x \in X,\\
&\tau_\varphi = \tau.
\end{aligned}$$

Označimo familiju $(A_\varphi)^D = \{(\sigma_\varphi)_u^D\}_{u \in X^*}$ sa $A_\varphi^D = \{\varphi_u^D\}_{u \in X^*}$, kao i fazi automat $(\mathcal{A}_\varphi)^D = ((A_\varphi)^D, \{(\sigma_\varphi)_\varepsilon^D / g(\sigma \circ \varphi)\}, (\delta_\varphi)^D, (\tau_\varphi)^D)$ jednostavno sa $\mathcal{A}_\varphi^D = (A_\varphi^D, \{\varphi_\varepsilon^D / g(\sigma \circ \varphi)\}, \delta_\varphi^D, \tau_\varphi^D)$. Očigledno je da se elementi familije A_φ^D mogu direktno zadati formulama:

$$\begin{aligned}
&\varphi_\varepsilon^D = f(\sigma \circ \varphi),\\
&\varphi_{ux}^D = f(\varphi_u^D \circ \delta_x \circ \varphi), \quad \text{za svako } u \in X^* \text{ i } x \in X,
\end{aligned}$$

dok se fazi relacija $\delta_\varphi^D \in L^{A_\varphi^D \times X \times A_\varphi^D}$ može zadati sa:

$$\delta_\varphi^D(\alpha, x, \beta) = \begin{cases} g(\varphi_u^D \circ \delta_x \circ \varphi), & \alpha = \varphi_u^D \text{ i } \beta = \varphi_{ux}^D, \text{ za neko } u \in X^* \text{ i } x \in X \\ 0, & \text{inače} \end{cases},$$

kao i fazi skup $\tau_\varphi^D \in L^{A_\varphi^D}$ sa:

$$\tau_\varphi^D(\alpha) = \alpha \circ \tau,$$

za svako $\alpha \in A_\varphi^D$. U slučaju kada je $\varphi = \triangle_A$ identička fazi relacija, tada je $\mathcal{A}_\varphi^D = \mathcal{A}^D$.

Prema Teoremi 4.7, $\mathcal{A}_\varphi^D$ je KDFA, stoga je fazi jezik raspoznat fazi automatom $\mathcal{A}_\varphi^D$, prema (2.12), jednak:

$$[\![\mathcal{A}_\varphi^D]\!](u) = c_u^{\varphi, D} \otimes \tau_\varphi^D(\varphi_u^D),$$

pri čemu je $c_u^{\varphi,D} \in \mathcal{L}$ jednako:

$$c_u^{\varphi,D} = g(\sigma \circ \varphi) \otimes \bigotimes_{i=1}^{n} \delta_\varphi^D(\varphi^D_{x_1x_2\ldots x_{i-1}}, x_i, \varphi^D_{x_1x_2\ldots x_i}). \tag{4.23}$$

za svako $u = x_1x_2\ldots x_n \in X^*$. Štaviše, posmatrajući fazi automat $\mathcal{A}_\varphi$ umesto $\mathcal{A}$ u Lemi 4.8, (4.20) postaje:

$$\sigma \circ \varphi \circ \bigcirc_{i=1}^{n} (\delta_{x_i} \circ \varphi) = c^{\varphi,D}_{x_1x_2\ldots x_n} \otimes \varphi^D_{x_1x_2\ldots x_n}. \tag{4.24}$$

Pokazaćemo da se, izborom odgovarajuće faktorizacije D od A i fazi relacije φ na A, konstrukcijom $\mathcal{A}_\varphi$ pre konstrukcije $\mathcal{A}^D$ dobijaju poboljšanja u odnosu na direktnu konstrukciju $\mathcal{A}^D$ iz $\mathcal{A}$. U tom svetlu, najpre određujemo uslove pod kojima su fazi automati $\mathcal{A}$ i $\mathcal{A}^D_\varphi$ ekvivalentni.

Teorema 4.12. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ refleksivna slabo desno invarijantna fazi relacija na A, i neka je $\mathcal{A}^D_\varphi$ KDFA od $\mathcal{A}$. Tada je $[\![\mathcal{A}]\!] = [\![\mathcal{A}^D_\varphi]\!]$.*

Dokaz. Odaberimo proizvoljnu reč $u = x_1x_2\ldots x_n \in X^*$ dužine $n \in \mathbb{N}_0$. Prema (2.9) i sukcesivne primene (3.11) sledi:

$$[\![\mathcal{A}_\varphi]\!](u) = \sigma \circ \varphi \circ \left(\bigcirc_{i=1}^{n} (\delta_{x_i} \circ \varphi) \right) \circ \tau = \sigma \circ \left(\bigcirc_{i=1}^{n} \delta_{x_i} \right) \circ \tau = [\![\mathcal{A}]\!](u).$$

S obzirom da je reč $u \in X^*$ odabrana na proizvoljan način, ekvivalencija fazi automata $\mathcal{A}_\varphi$ i $\mathcal{A}$ sledi. Ali iz Teoreme 4.7 znamo da su fazi automati $\mathcal{A}_\varphi$ i $\mathcal{A}^D_\varphi$ ekvivalentni. Ovim je dokaz Teoreme završen. □

Sada pokazujemo da se veličine odgovarajućih KDFA-a od $\mathcal{A}$ mogu uporediti kada se koriste *desno invarijantna fazi kvazi-uređenja* i *maksimalne faktorizacije*.

Teorema 4.13. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ maksimalna faktorizacija od A, i neka su φ i ϕ desno invarijantna fazi kvazi-uređenja na A. Neka su $\mathcal{A}^D_\varphi$ i $\mathcal{A}^D_\phi$ odgovarajući KDFA-i od $\mathcal{A}$. Ako je $\varphi \leq \phi$, tada sledi $|\mathcal{A}^D_\phi| \leq |\mathcal{A}^D_\varphi|$.*

Dokaz. Definišimo preslikavanje $\xi : A^D_\varphi \to A^D_\phi$ sa $\xi(\varphi^D_u) = \phi^D_u$, za svako $u \in X^*$. Tada je ξ dobro definisano preslikavanje. Zaista, neka su $u = x_1x_2\ldots x_n \in X^*$ i $v = y_1y_2\ldots y_m \in X^*$ dve reči za koje je $\varphi^D_u = \varphi^D_v$. Tada iz (4.24) dobijamo:

$$\begin{aligned} c^{\varphi,D}_v \otimes \left(\sigma \circ \varphi \circ \bigcirc_{i=1}^{n} (\delta_{x_i} \circ \varphi) \right) &= c^{\varphi,D}_v \otimes c^{\varphi,D}_u \otimes \varphi^D_u \\ &= c^{\varphi,D}_u \otimes c^{\varphi,D}_v \otimes \varphi^D_v \\ &= c^{\varphi,D}_u \otimes \left(\sigma \circ \varphi \circ \bigcirc_{i=1}^{m} (\delta_{y_i} \circ \varphi) \right). \end{aligned}$$

Koristeći prethodni identitet u kombinaciji sa Lemom 1.24, zaključujemo da važi:

$$\begin{aligned} &c^{\varphi,D}_v \otimes c^{\phi,D}_u \otimes \phi^D_u \\ &= c^{\varphi,D}_v \otimes \left(\sigma \circ \phi \circ \bigcirc_{i=1}^{n} (\delta_{x_i} \circ \phi) \right) = c^{\varphi,D}_v \otimes (\sigma \circ \delta_u \circ \phi) = c^{\varphi,D}_v \otimes (\sigma \circ \delta_u \circ \varphi \circ \phi) \end{aligned}$$

$$= c_v^{\varphi,D} \otimes \left(\sigma \circ \varphi \circ \left(\bigcirc_{i=1}^{n} (\delta_{x_i} \circ \varphi) \right) \circ \phi \right) = \left(c_v^{\varphi,D} \otimes \left(\sigma \circ \varphi \circ \bigcirc_{i=1}^{n} (\delta_{x_i} \circ \varphi) \right) \right) \circ \phi$$

$$= \left(c_u^{\varphi,D} \otimes \left(\sigma \circ \varphi \circ \bigcirc_{i=1}^{m} (\delta_{y_i} \circ \varphi) \right) \right) \circ \phi = c_u^{\varphi,D} \otimes \left(\sigma \circ \varphi \circ \left(\bigcirc_{i=1}^{m} (\delta_{y_i} \circ \varphi) \right) \circ \phi \right)$$

$$= c_u^{\varphi,D} \otimes (\sigma \circ \delta_v \circ \varphi \circ \phi) = c_u^{\varphi,D} \otimes (\sigma \circ \delta_v \circ \phi) = c_u^{\varphi,D} \otimes \left(\sigma \circ \phi \circ \bigcirc_{i=1}^{m} (\delta_{y_i} \circ \phi) \right).$$

Ponovo, iz (4.24) dobijamo:

$$c_v^{\varphi,D} \otimes c_u^{\phi,D} \otimes \phi_u^D = c_u^{\varphi,D} \otimes c_v^{\phi,D} \otimes \phi_v^D,$$

te s obzirom da je D maksimalna faktorizacija od A, primenjujući (4.21) i (4.11) dobijamo:

$$\phi_u^D = f(\phi_u^D) = f(c_v^{\varphi,D} \otimes c_u^{\phi,D} \otimes \phi_u^D) = f(c_u^{\varphi,D} \otimes c_v^{\phi,D} \otimes \phi_v^D) = f(\phi_v^D) = \phi_v^D.$$

U zaključku, dokazali smo da $\varphi_u^D = \varphi_v^D$ povlači $\phi_u^D = \phi_v^D$, što obezbeđuje da je ξ dobro definisano preslikavanje. Štaviše, očigledno je da za svako $\phi_u^D \in A_\phi^D$ postoji $\varphi_u^D \in A_\varphi^D$ tako da je $\xi(\varphi_u^D) = \phi_u^D$, za proizvoljno $u \in X^*$, te je stoga ξ i surjektivno preslikavanje. Dakle, $|A_\phi^D| \leq |A_\varphi^D|$ sledi. □

Napomena 4.14. Prethodna Teorema navodi da se korišćenjem najvećeg desno invarijantnog fazi kvazi-uređenja na A dobija odgovarajući KDFA od $\mathcal{A}$ koji nema veći broj stanja od KDFA od $\mathcal{A}$ dobijenog korišćenjem proizvoljnog desno invarijantnog fazi kvazi-uređenja na A i, posebno, korišćenjem fazi identičke relacije na A, pod uslovom da se takođe koristi i maksimalna faktorizacija od A. Smanjenje broja stanja može biti značajno, odnosno može se dobiti KDFA sa konačnim brojem stanja, kao što je pokazano primerima u Sekciji 4.6.

Sada ćemo pokazati da ekvivalencija fazi automata $\mathcal{A}$ i $\mathcal{A}_\varphi^D$ važi i u slučaju kada je φ refleksivna levo invarijantna fazi relacija na A. Ipak, korišćenje refleksivnih levo invarijantnih fazi relacija na A ne daje poboljšanja u konstrukciji KDFA-a od $\mathcal{A}$.

Lema 4.15. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ refleksivna slabo levo invarijantna fazi relacija na A, i neka je $\mathcal{A}_\varphi^D$ KDFA od $\mathcal{A}$. Tada za svako $n \in \mathbb{N}_0$ i $u = x_1 x_2 \dots x_n \in X^*$ važi:*

$$\sigma_u = \sigma \circ \delta_u = c_u^{\varphi,D} \otimes \varphi_u^D, \tag{4.25}$$

pri čemu je $c_u^{\varphi,D}$ definisano sa (4.23).

Dokaz. Dokaz ide preko indukcije po dužini n reči u. U slučaju kada je $n = 0$ imamo:

$$\sigma_\varepsilon = \sigma = \sigma \circ \varphi = g(\sigma \circ \varphi) \otimes f(\sigma \circ \varphi) = \sigma_\varphi^D(\varphi_\varepsilon^D) \otimes \varphi_\varepsilon^D$$

Drugim rečima, (4.25) važi kada je $n = 0$. Pretpostavimo sada da (4.25) važi za neko $n \in \mathbb{N}$ i simbole $x_1, x_2, \dots, x_n \in X$. Neka je $x_{n+1} \in X$. Tada važi:

$$\sigma \circ \delta_{x_1 x_2 \dots x_{n+1}} = \sigma \circ \delta_{x_1 x_2 \dots x_{n+1}} \circ \varphi = \sigma \circ \delta_{x_1 x_2 \dots x_n} \circ \delta_{x_{n+1}} \circ \varphi$$

Koristeći slične principe kao u Lemi 4.8 dobijamo:

$$\sigma \circ \delta_u = c_{x_1 x_2 \dots x_n}^{\varphi,D} \otimes (\varphi_{x_1 x_2 \dots x_n}^D \circ \delta_{x_{n+1}} \circ \varphi)$$

$$
\begin{aligned}
&= c^{\varphi,D}_{x_1x_2\ldots x_n} \otimes \big(g(\varphi^D_{x_1x_2\ldots x_n} \circ \delta_{x_{n+1}} \circ \varphi) \otimes f(\varphi^D_{x_1x_2\ldots x_n} \circ \delta_{x_{n+1}} \circ \varphi)\big) \\
&= \big(c^{\varphi,D}_{x_1x_2\ldots x_n} \otimes \delta^D_\varphi(\varphi^D_{x_1x_2\ldots x_n}, x_n, \varphi^D_{x_1x_2\ldots x_{n+1}})\big) \otimes \varphi^D_{x_1x_2\ldots x_{n+1}} \\
&= c^{\varphi,D}_{x_1x_2\ldots x_nx_{n+1}} \otimes \varphi^D_{x_1x_2\ldots x_{n+1}},
\end{aligned}
$$

čime je dokaz završen. □

Teorema 4.16. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ refleksivna slabo levo invarijantna fazi relacija na A, i neka je $\mathcal{A}^D_\varphi$ KDFA od $\mathcal{A}$. Tada je $[\![\mathcal{A}]\!] = [\![\mathcal{A}^D_\varphi]\!]$.*

Dokaz. Prema (2.9) i (4.25), za svako $u \in X^*$ važi:

$$[\![\mathcal{A}]\!](u) = \sigma \circ \delta_u \circ \tau = c^{\varphi,D}_u \otimes (\varphi^D_u \circ \tau).$$

S obzirom da je $\varphi^D_u \circ \tau = \tau^D_\varphi(\varphi^D_u)$, iz (4.18) dobijamo $[\![\mathcal{A}]\!](u) = [\![\mathcal{A}^D_\varphi]\!](u)$, čime je dokaz završen. □

Teorema 4.17. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ maksimalna faktorizacija od A, φ i ϕ refleksivne slabo levo invarijantne fazi relacije na A, i neka su $\mathcal{A}^D_\varphi$ i $\mathcal{A}^D_\phi$ odgovarajući KDFA-i od $\mathcal{A}$. Tada važi $|\mathcal{A}^D_\phi| = |\mathcal{A}^D_\varphi| = |\mathcal{A}^D|$.*

Dokaz. Iz (4.25) sledi da za svako $u \in X^*$ važi:

$$\sigma \circ \delta_u = c^{\varphi,D}_u \otimes \varphi^D_u = c^{\phi,D}_u \otimes \phi^D_u.$$

Koristeći činjenicu da je D maksimalna faktorizacija od A, dobijamo:

$$\varphi^D_u = f(\varphi^D_u) = f(c^{\varphi,D}_u \otimes \varphi^D_u) = f(\sigma \circ \delta_u) = f(c^{\phi,D}_u \otimes \phi^D_u) = f(\phi^D_u) = \phi^D_u.$$

Prethodni identitet važi za svako $u \in X^*$, stoga je preslikavanje $\xi : A^D_\varphi \to A^D_\phi$, definisano sa $\xi(\varphi^D_u) = \phi^D_u$, za svako $u \in X^*$, dobro definisano i bijektivno. □

Napomena 4.18. Prethodna Teorema pokazuje da korišćenje refleksivnih slabo levo invarijantnih fazi relacija ne pruža poboljšanje u konstrukciji KDFA-a u odnosu na korišćenje identičke fazi relacije kada se koriste *maksimalne faktorizacije*. Ali, refleksivne slabo levo invarijantne fazi relacije daju poboljšanje kada se koriste proizvoljne faktorizacije, kao što je pokazano primerima u Sekciji 4.6.

4.3 Determinizacija preko konstrukcije dečjeg fazi automata

U ovoj sekciji, pokazujemo da se dodatna poboljšanja mogu dobiti konstrukcijom takozvanog *dečjeg fazi automata*. Neka je $\mathcal{L}$ kompletna reziduirana mreža koja je bez delioca nule i zadovoljava svojstvo deljivosti, i neka je $X = \{x_1, x_2, \ldots, x_m\}$ alfabet. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat nad $\mathcal{L}$ i X, $D = (f, g)$ faktorizacija od A, i neka je φ fazi relacija na A. Za svako $u \in X^*$ definišimo $(2m+1)$-torku $(\varphi^D_u)^c \in (L^A \times L)^m \times L$ sa

$$(\varphi^D_u)^c = (\varphi^D_{ux_1}, g(\varphi^D_u \circ \delta_{x_1} \circ \varphi), \varphi^D_{ux_2}, g(\varphi^D_u \circ \delta_{x_2} \circ \varphi), \ldots, \varphi^D_{ux_m}, g(\varphi^D_u \circ \delta_{x_m} \circ \varphi), \varphi^D_u \circ \tau). \tag{4.26}$$

Dalje, neka je $(A_\varphi^D)^c = \{(\varphi_u^D)^c | u \in X^*\}$, te definišimo preslikavanja $(\delta_\varphi^D)^c : (A_\varphi^D)^c \times X \times (A_\varphi^D)^c \to L$ i $(\tau_\varphi^D)^c : (A_\varphi^D)^c \to L$ sa:

$$(\delta_\varphi^D)^c(\alpha^c, x, \beta^c) = \begin{cases} g(\varphi_u^D \circ \delta_x \circ \varphi), & \alpha = \varphi_u^D \text{ i } \beta = \varphi_{ux}^D, \text{ za neko } u \in X^* \text{ i } x \in X, \\ 0, & \text{inače} \end{cases},$$

$$(\tau_\varphi^D)^c(\alpha^c) = \alpha \circ \tau,$$

za svako $\alpha, \beta \in A_\varphi^D$. Tada važi sledeće tvrđenje.

Lema 4.19. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, i φ fazi relacija na A. Tada je $(\mathcal{A}_\varphi^D)^c = ((A_\varphi^D)^c, \{(\varphi_\varepsilon^D)^c / g(\sigma \circ \varphi)\}, (\delta_\varphi^D)^c, (\tau_\varphi^D)^c)$ dobro definisani KDFA.*

Dokaz. (Dobro definisan) Najpre pokazujemo da su $(\delta_\varphi^D)^c$ i $(\tau_\varphi^D)^c$ dobro definisana preslikavanja. Neka je $(\varphi_u^D)^c = (\varphi_v^D)^c$ za neko $u, v \in X^*$. Tada iz definicije (4.26) direktno sledi da je $\varphi_{ux_i}^D = \varphi_{vx_i}^D$ i $g(\varphi_u^D \circ \delta_{x_i} \circ \varphi) = g(\varphi_v^D \circ \delta_{x_i} \circ \varphi)$ za svako $1 \leq i \leq m$, kao i $\varphi_u^D \circ \tau = \varphi_v^D \circ \tau$. Odaberimo proizvoljno $x \in X$. Tada je $\varphi_{ux}^D = \varphi_{vx}^D$, te zbog toga:

$$\varphi_{uxx_i}^D = f(\varphi_{ux}^D \circ \delta_{x_i} \circ \varphi) = f(\varphi_{vx}^D \circ \delta_{x_i} \circ \varphi) = \varphi_{vxx_i}^D$$

za svako $1 \leq i \leq m$, kao i $\varphi_{ux}^D \circ \tau = \varphi_{vx}^D \circ \tau$. Stoga, $(\varphi_{ux}^D)^c = (\varphi_{vx}^D)^c$. Štaviše, takođe iz definicije (4.26) imamo da je $g(\varphi_u^D \circ \delta_x \circ \varphi) = g(\varphi_v^D \circ \delta_x \circ \varphi)$, stoga je:

$$(\delta_\varphi^D)^c((\varphi_u^D)^c, x, (\varphi_{ux}^D)^c) = g(\varphi_u^D \circ \delta_x \circ \varphi) = g(\varphi_v^D \circ \delta_x \circ \varphi) = (\delta_\varphi^D)^c((\varphi_v^D)^c, x, (\varphi_{vx}^D)^c).$$

Ovo znači da je $(\delta_\varphi^D)^c$ dobro definisano preslikavanje. Takođe, lako se proverava da je i $(\tau_\varphi^D)^c$ takođe dobro definisano preslikavanje, čime je dokazano da je $(\mathcal{A}_\varphi^D)^c$ dobro definisan fazi automat.
(Kompletan) Prema konstrukciji fazi automata $(\mathcal{A}_\varphi^D)^c$, imamo da je $(\varphi_u^D)^c \in (A_\varphi^D)^c$ i $(\varphi_{ux}^D)^c \in (A_\varphi^D)^c$ za svako $u \in X^*$ i $x \in X$, stoga prema svojstvu (4.3) imamo da važi:

$$(\delta_\varphi^D)^c((\varphi_u^D)^c, x, (\varphi_{ux}^D)^c) = g(\varphi_u^D \circ \delta_x \circ \varphi) > 0.$$

(Deterministički) Prema konstrukciji fazi automata $(\mathcal{A}_\varphi^D)^c$, φ_ε^D je jedinstveno početno stanje sa stepenom $g(\sigma \circ \varphi)$. Takođe, ako je $(\delta_\varphi^D)^c((\varphi_u^D)^c, x, \beta_1) > 0$ i $(\delta_\varphi^D)^c((\varphi_u^D)^c, x, \beta_2) > 0$, za neko $u \in X^*$, tada je prema konstrukciji $(\mathcal{A}_\varphi^D)^c$, $\beta_1 = \beta_2 = (\varphi_{ux}^D)^c$. □

U slučaju kada je $D = N$ trivijalna faktorizacija, možemo se uveriti da za svako $u \in X^*$ $(2m+1)$-torka $(\varphi_u^N)^c \in (A_\varphi^N)^c$ postaje

$$(\varphi_u^N)^c = (\varphi_u^N \circ \delta_{x_1} \circ \varphi, 1, \varphi_u^N \circ \delta_{x_2} \circ \varphi, 1, \ldots, \varphi_u^N \circ \delta_{x_m} \circ \varphi, 1, \varphi_u^N \circ \tau),$$

te se stoga može identifikovati sa $(m+1)$-torkom $(\varphi_u)^c \in (L^A)^m \times L$ definisanoj u [110] sa

$$(\varphi_u)^c = (\varphi_u^N \circ \delta_{x_1} \circ \varphi, \varphi_u^N \circ \delta_{x_2} \circ \varphi, \ldots, \varphi_u^N \circ \delta_{x_m} \circ \varphi, \varphi_u^N \circ \tau).$$

Zbog toga, $(\mathcal{A}_\varphi^D)^c$ predstavlja generalizaciju fazi automata konstruisanih u [109, 110]. Pošto su ti fazi automati u [109, 110] nazvani dečji fazi automati Nerodovog fazi automata od $\mathcal{A}$, KDFA $\mathcal{A}_D^c$ nazivamo *kompletan deterministički dečji fazi automat (KDDFA)* od fazi automata $\mathcal{A}$. Takođe, kada je D proizvoljna faktorizacija i $\varphi = \triangle_A$, tada odgovarajući KDDFA od $\mathcal{A}$ označavamo sa $(\mathcal{A}^D)^c = ((A^D)^c, \{(\sigma_\varepsilon^D)^c / g(\sigma)\}, (\delta^D)^c, (\tau^D)^c)$, gde je $(A^D)^c = \{(\sigma_u^D)^c | u \in X^*\}$.

Sada pronalazimo uslove pod kojima su fazi automati $(\mathcal{A}^D)^c$ i $\mathcal{A}$ ekvivalentni.

Teorema 4.20. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ refleksivna slabo desno invarijantna ili refleksivna slabo levo invarijantna fazi relacija na A, i neka je $(\mathcal{A}_\varphi^D)^c$ KDDFA od $\mathcal{A}$. Tada je $[\![\mathcal{A}]\!] = [\![(\mathcal{A}_\varphi^D)^c]\!]$.*

Dokaz. Prema definiciji fazi automata $(\mathcal{A}_\varphi^D)^c$, zaključujemo da je

$$(\delta_\varphi^D)^c((\varphi_u^D)^c, x, (\varphi_{ux}^D)^c) = \delta_\varphi^D(\varphi_u^D, x, \varphi_{ux}^D),$$

kao i

$$(\tau_\varphi^D)^c((\varphi_u^D)^c) = \tau_\varphi^D(\varphi_u^D),$$

za svako $u \in X^*$. Stoga, prema Lemi 4.19, $(\mathcal{A}_\varphi^D)^c$ je KDFA, te za svako $u = x_1x_2\dots x_n \in X^*$ imamo da je:

$$\begin{aligned}
[\![(\mathcal{A}_\varphi^D)^c]\!](u) &= g(\sigma \circ \varphi) \otimes \left(\bigotimes_{i=1}^{n} (\delta_\varphi^D)^c((\varphi_{x_1x_2\dots x_{i-1}}^D)^c, x_i, (\varphi_{x_1x_2\dots x_i}^D)^c)\right) \otimes (\tau_\varphi^D)^c((\varphi_u^D)^c) \\
&= g(\sigma \circ \varphi) \otimes \left(\bigotimes_{i=1}^{n} \delta_\varphi^D(\varphi_{x_1x_2\dots x_{i-1}}^D, x_i, \varphi_{x_1x_2\dots x_i}^D)\right) \otimes \tau_\varphi^D(\varphi_u^D) \\
&= [\![\mathcal{A}_\varphi^D]\!](u).
\end{aligned}$$

Drugim rečima, $(\mathcal{A}_\varphi^D)^c$ je ekvivalentan sa $\mathcal{A}_\varphi^D$, za proizvoljnu faktorizaciju D od A i proizvoljnu fazi relaciju φ na A. Ali znamo da je $\mathcal{A}_\varphi^D$ ekvivalentan sa $\mathcal{A}$ kada je φ refleksivna slabo desno invarijantna ili refleksivna slabo levo invarijantna fazi relacija na A, čime je dokaz završen. □

Sada pokazujemo prednost konstrukcije dečjeg fazi automata, ili preciznije, da dečiji fazi automat $(\mathcal{A}_\varphi^D)^c$ ne može imati veći broj stanja od fazi automata $\mathcal{A}_\varphi^D$.

Teorema 4.21. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ fazi relacija na A, $\mathcal{A}_\varphi^D$ KDFA od $\mathcal{A}$ i $(\mathcal{A}_\varphi^D)^c$ KDDFA od $\mathcal{A}$. Tada je $|(\mathcal{A}_\varphi^D)^c| \leq |\mathcal{A}_\varphi^D|$.*

Dokaz. Neka je $\xi : A_\varphi^D \to (A_\varphi^D)^c$ preslikavanje definisano sa:

$$\xi(\varphi_u^D) = (\varphi_u^D)^c,$$

za svako $u \in X^*$. Tada je preslikavanje ξ dobro definisano. Zaista, ako za neke reči $u, v \in X^*$ imamo da važi $\varphi_u^D = \varphi_v^D$, tada takođe važi:

$$\begin{aligned}
\xi(\varphi_u^D) &= (\varphi_u^D)^c = (\varphi_{ux_1}^D, g(\varphi_u^D \circ \delta_{x_1} \circ \varphi), \dots, \varphi_{ux_m}^D, g(\varphi_u^D \circ \delta_{x_m} \circ \varphi), \varphi_u^D \circ \tau) \\
&= (f(\varphi_u^D \circ \delta_{x_1} \circ \varphi), g(\varphi_u^D \circ \delta_{x_1} \circ \varphi), \dots, f(\varphi_u^D \circ \delta_{x_m} \circ \varphi), g(\varphi_u^D \circ \delta_{x_m} \circ \varphi), \varphi_u^D \circ \tau) \\
&= (f(\varphi_v^D \circ \delta_{x_1} \circ \varphi), g(\varphi_v^D \circ \delta_{x_1} \circ \varphi), \dots, f(\varphi_v^D \circ \delta_{x_m} \circ \varphi), g(\varphi_v^D \circ \delta_{x_m} \circ \varphi), \varphi_v^D \circ \tau) \\
&= (\varphi_{vx_1}^D, g(\varphi_v^D \circ \delta_{x_1} \circ \varphi), \dots, \varphi_{vx_m}^D, g(\varphi_v^D \circ \delta_{x_m} \circ \varphi), \varphi_v^D \circ \tau) = (\varphi_v^D)^c \\
&= \xi(\varphi_v^D).
\end{aligned}$$

Jasno je i da je ξ surjektivno preslikavanje, čime je dokaz Teoreme završen. □

Teorema 4.22. *Neka je $\mathcal{A}$ fazi automat, $D = (f, g)$ faktorizacija od A, φ fazi relacija na A, $\mathcal{A}_\varphi^D$ KDFA od $\mathcal{A}$ i $(\mathcal{A}_\varphi^D)^c$ KDDFA od $\mathcal{A}$. Tada je $\mathcal{A}_\varphi^D$ konačan akko je $(\mathcal{A}_\varphi^D)^c$ konačan.*

Dokaz. Najpre, pretpostavimo da je $(A_\varphi^D)^c$ konačan skup, tj. $(A_\varphi^D)^c = \{(\varphi_{u_1}^D)^c, (\varphi_{u_2}^D)^c, \dots, (\varphi_{u_k}^D)^c\}$ za neko $k \in \mathbb{N}$ i $u_1, \dots, u_k \in X^*$. Za svako $u \in X^+$ imamo da je $u =$

vx za neko $v \in X^*$ i $x \in X$, kao i $(\varphi_v^D)^c = (\varphi_{u_j}^D)^c$ za neko $j \in \{1, \ldots, k\}$. Ovo poslednje podrazumeva da je $\varphi_{vy}^D = \varphi_{u_j y}^D$ za svako $y \in X$, ili $f(\varphi_v^D \circ \delta_y \circ \varphi) = f(\varphi_{u_j}^D \circ \delta_y \circ \varphi)$ za svako $y \in X$. Stoga dobijamo da je $\varphi_u^D = f(\varphi_v^D \circ \delta_x \circ \varphi) = f(\varphi_{u_j}^D \circ \delta_x \circ \varphi) = \varphi_{u_j x}^D$. Dakle, $A_\varphi^D \backslash \{\varphi_\varepsilon^D\} = \{\varphi_{u_j x_l}^D | 1 \leq j \leq k,\ 1 \leq l \leq m\}$, te je stoga $|A_\varphi^D| \leq km + 1$, osnosno, $|A_\varphi^D| \leq |(A_\varphi^D)^c| \cdot |X| + 1$. Drugi smer tvrđenja direktno sledi iz Teoreme 4.21. □

4.4 Algoritmi za determinizaciju fazi automata i njihova analiza

U ovoj sekciji dajemo algoritme za konstrukciju grafova prelaza KDFA-a prikazanih u prethodnim sekcijama. Koristimo dobro poznate strukture podataka i operacije na njima. Preciznije, koristimo *red* kao strukturu podataka snabdevenu sledećim standardnim operacijama:

- `Enqueue`(q, v), koja dodaje stavku v na kraj reda q,
- `Dequeue`(q), koja uklanja stavku sa početka reda q i vraća je kao rezultat,
- `IsEmpty`(q), koja vraća `true` ukoliko je red q prazan i ne postoji stavka koja se može uzeti sa početka reda, i `false` inače.

Sve navedene operacije izvršavaju se u $\mathcal{O}(1)$ vremenu.

Dalje, koristimo *stablo* kao strukturu podataka snabdevenu sledećim standardnim operacijama:

- `Lookup`(T, v), koja vraća `true` ako postoji čvor sa vrednošću v u stablu T, i `false` inače,
- `Insert`(T, n, v, l), koja dodaje novi čvor sa vrednošću n u stablu T sa granom označenom sa l iz postojećeg čvora sa vrednošću v,
- `GetValue`(T, v), koja vraća čvor sa vrednošću v u stablu T.

Takođe koristimo i oznaku `Insert`$(T, n) =$ `Insert`$(T, n, null, null)$ za dodavanje čvora sa vrednošću n kao koren praznog stabla T. Označimo sa t broj čvorova stabla T, a sa c cenu računanja da li su vrednosti dva čvora stabla T jednake. Tada operacija `Lookup`(T, v) zahteva, u najgorem slučaju, prolazak kroz sve čvorove stabla T (što zahteva $\mathcal{O}(t)$ vreme), i za se za svaki čvor proveru da li je njegova vrednost jednaka zadatoj vrednosti v (što zahteva $\mathcal{O}(c)$ vreme). Dakle, ukupno vreme izrvšenja operacije `Lookup` je $\mathcal{O}(tc)$. Na sličan način možemo zaključiti da se i operacije `Insert` i `GetValue` izvršavaju u $\mathcal{O}(tc)$ vremenu.

Na kraju, kao strukturu podataka koristimo i *usmereni graf* $G = (V, E)$ snabdevenu sledećim standardnim operacijama:

- `AddVertex`(G, v), koja kreira novi čvor sa vrednošću v u grafu G,
- `AddEdge`(G, u, v, e), koja kreira usmerenu granu sa oznakom e iz čvora sa vrednošću u do čvora sa vrednošću v u grafu G.

Kada je graf predstavljen preko liste susedstva, tada se obe operacije izvršavaju u $\mathcal{O}(1)$ vremenu, a kada je graf predstavljen preko matrice susedstva, tada se operacija `AddEdge` izvršava u $\mathcal{O}(1)$, dok se operacija `AddVertex` izvršava u $\mathcal{O}(|V|^2)$ vremenu. Takođe, koristimo i operacije:

- `Lookup`(G, v), koja vraća `true` ako postoji čvor sa vrednošći v u grafu G, i `false` inače,

- `GetValue`(G, v), koja vraća čvor sa vrednošću v u grafu G.

Slično, ako se sa c označi trošak provere jednakosti vrednosti dva čvora u grafu G, tada se obe operacije `Lookup` i `GetValue` izvršavaju u $\mathcal{O}(|V|c)$ vremenu.

Najpre dajemo algoritam za konstrukciju KDFA od fazi automata $\mathcal{A}$. Preciznije, za dati fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ gde je $|A| = n$ i $|X| = m$, faktorizaciju $D = (f, g)$ od A, i datu fazi relaciju φ na A, cilj je konstrukcija KDFA $\mathcal{A}_\varphi^D = (A_\varphi^D, \{\varphi_\varepsilon^D / g(\sigma \circ \varphi)\}, \delta_\varphi^D, \tau_\varphi^D)$ od $\mathcal{A}$. Dati algoritam oslanja se na teoretske rezultate iz prethodnih sekcija, i njegov formalni opis dat je Algoritmom 6. Ideja Algoritma 6 je da se induktivno konstruiše puno m-arno *stablo prelaza* od $\mathcal{A}_\varphi^D$ čiji su čvorovi fazi skupovi. Da bismo konstruisali stablo prelaza, koristimo pomoćni red čiji su elementi fazi skupovi. Zapravo, koristimo red za skladištenje čvorova iz stabla prelaza koje konstruišemo. Primetimo da se, zbog korišćenja reda kao strukture podataka, stablo prelaza od $\mathcal{A}_\varphi^D$ konstruiše u BFS (Breadth-first search) maniru, odnosno, stanja koja su dostižna iz početnog stanja pod dejstvom kraćih reči generisana su i procesirana pre stanja koja su dostižna iz početnog stanja pod dejstvom dužih reči. Takođe, koristimo i pokazivače $s(\cdot)$ koji pridružuju čvorovima stablu prelaza koje konstruišemo odgovarajuće cele brojeve, koje kasnije koristimo kod kreiranja grafa prelaza od $\mathcal{A}_\varphi^D$ iz njegovog stabla prelaza. Induktivna procedura za konstrukciju grafa prelaza od $\mathcal{A}_\varphi^D$ data je Algoritmom 6.

Procedura za konstruisanje grafa prelaza KDFA od $\mathcal{A}$ ne mora da se završi u konačnom broju koraka, zbog činjenice da kolekcija fazi podskupova $\{\varphi_u^D\}_{u \in X^*}$ može biti beskonačna. Međutim, u slučajevima kada je ova kolekcija konačna, procedura se završava u konačnom broju koraka, nakon izračunavanja svih elemenata te kolekcije. Kao što će biti dokazano u Sekciji 4.5, kada je D maksimalna faktorizacija od A i φ desno invarijantno fazi kvazi-uređenje na A, tada se procedura završava akko fazi automat $\mathcal{A}_\varphi$ zadovoljava takozvano *slabo svojstvo reprezentativnih ciklusa*. U tom slučaju, označimo sa k broj različitih vrednosti koje fazi podskupovi iz familije $\{\varphi_u^D\}_{u \in X^*}$ mogu uzeti. Tada se iz ovih k različitih vrednosti mogu konstruisati najviše k^n različitih fazi podskupova od A, što znači da kolekcija $\{\varphi_u^D\}_{u \in X^*}$ može imati najviše k^n različitih elemenata.

Sada analiziramo vreme izvršenja Algoritma 6. Označimo sa $c_\vee$, $c_\wedge$, $c_\otimes$, $c_\to$, c_g i c_f, redom, trošak izvršenja operacija $\vee, \wedge, \otimes, \to$ u $\mathcal{L}$, i operacija g i f faktorizacije D od A. Na primer, kada je $\mathcal{L}$ standardna proizvod, standardna Gödelova ili standardna Łukasiewiczeva struktura, tada imamo da je $c_\vee = c_\wedge = c_\otimes = c_\to = 1$, a kada je $D = (f, g) = D_M(f_M, g_M)$ Mohrijeva faktorizacija, tada je $c_g = c_f = n$. Podsetimo se da je $|\mathcal{A}| = n$ i $|X| = m$.

Analiziramo samo one korake iz Algoritma 6 čije je vreme izvršenja veće od $\mathcal{O}(1)$. Računanje $f(\sigma \circ \varphi)$ u koraku 3 najpre zahteva računanje kompozicije $\sigma \circ \varphi$ (koje se završava u $\mathcal{O}(n^2(c_\otimes + c_\vee))$ vremenu), te zatim računanje $f(\sigma \circ \varphi)$ (što se završava u $\mathcal{O}(c_f)$ vremenu), što daje ukupno vreme izvršenja $\mathcal{O}(n^2(c_\otimes + c_\vee) + c_f)$ za korak 3. Slično, prvo računanje u koraku 5 završava se u $\mathcal{O}(n^2(c_\otimes + c_\vee) + c_g)$ vremenu, a drugo u $\mathcal{O}(n(c_\otimes + c_\vee))$ vremenu.

Sada, vidimo da se korak 11 izvršava u $\mathcal{O}(n^2(c_\otimes + c_\vee) + c_f)$ vremenu, dok se korak 12 izvršava u $\mathcal{O}(n(c_\otimes + c_\vee))$ vremenu, odnosno, oba koraka zahtevaju polinomijalno vreme. S obzirom da stablo prelaza može imati najviše k^n čvorova, a provera da li su dva fazi skupa jednaka zahteva $\mathcal{O}(n)$ vremena, zaključujemo da svaki od koraka 13, 17 i 19 zahteva po $\mathcal{O}(nk^n)$ vremena, tj. svi zahtevaju eksponencijalno vreme. Stoga, koraci 11–19 izvršavaju se u $\mathcal{O}(nk^n)$ vremenu. Pošto je $|X| = m$, imamo da se petlja koja formira korake 10–20 izvršava u $\mathcal{O}(mnk^n)$ vremenu. Na kraju, imajući u vidu da su elementi reda q elementi kolekcije $\{\varphi_u^D\}_{u \in X^*}$, kao i da se preko koraka 13

Algoritam 6 Konstrukcija grafa prelaza kompletnog determinističkog fazi automata $\mathcal{A}_\varphi^D$

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad kompletnom reziduiranom mrežom $\mathcal{L}$ i alfabetom X, faktorizacija $D = (f, g)$ od A, i fazi relacija φ na A.
Izlaz: Kompletan deterministički fazi automat $\mathcal{A}_\varphi^D = (A_\varphi^D, \sigma_\varphi^D, \delta_\varphi^D, \tau_\varphi^D)$.

1: $\mathbf{S} \leftarrow 1$
2: Inicijalizovati prazan red q čiji su elementi fazi skupovi, kao i prazno stablo T ciji su čvorovi takođe fazi skupovi
3: $\varphi_\varepsilon^D \leftarrow f(\sigma \circ \varphi)$
4: $\texttt{Insert}(T, \varphi_\varepsilon^D)$
5: $\sigma_\varphi^D(\varphi_\varepsilon^D) \leftarrow g(\sigma \circ \varphi), \quad \tau_\varphi^D(\varphi_\varepsilon^D) \leftarrow \varphi_\varepsilon^D \circ \tau$
6: $s(\varphi_\varepsilon^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
7: $\texttt{Enqueue}(q, \varphi_\varepsilon^D)$
8: **while** not $\texttt{IsEmpty}(q)$ **do**
9: $\quad \varphi_u^D \leftarrow \texttt{Dequeue}(q)$
10: $\quad$ **for all** $x \in X$ **do**
11: $\quad\quad \varphi_{u\cdot x}^D \leftarrow f(\varphi_u^D \circ \delta_x \circ \varphi)$
12: $\quad\quad \sigma_\varphi^D(\varphi_{u\cdot x}^D) \leftarrow 0, \quad \tau_\varphi^D(\varphi_{u\cdot x}^D) \leftarrow \varphi_{u\cdot x}^D \circ \tau$
13: $\quad\quad$ **if** not $\texttt{Lookup}(T, \varphi_{u\cdot x}^D)$ **then**
14: $\quad\quad\quad \texttt{Enqueue}(q, \varphi_{u\cdot x}^D)$
15: $\quad\quad\quad s(\varphi_{u\cdot x}^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
16: $\quad\quad$ **else**
17: $\quad\quad\quad s(\varphi_{u\cdot x}^D) \leftarrow s(\texttt{GetValue}(T, \varphi_{u\cdot x}^D))$
18: $\quad\quad$ **end if**
19: $\quad\quad \texttt{Insert}(T, \varphi_{u\cdot x}^D, \varphi_u^D, x/g(\varphi_u^D \circ \delta_x \circ \varphi))$
20: $\quad$ **end for**
21: **end while**
22: Stablo prelaza T je konstruisano. Spajamo listove stabla T sa njegovim unutrašnjim čvorovima sa istom vrednošću pokazivača s. Rezultujući dijagram predstavlja graf prelaza od $\mathcal{A}_\varphi^D$.

obezbeđuje da se svaki element kolekcije nalazi samo jedanput u redu q u nekom trenutku, zaključujemo da se koraci 8–21 izvršavaju u $\mathcal{O}(mnk^{2n})$ vremenu. S obzirom da izvršenje koraka 22 ne prelazi ovo vreme, zaključujemo da je ukupno vreme izvršenja Algoritma 6 jednako $\mathcal{O}(mnk^{2n})$.

Sada dajemo algoritam za konstrukciju KDDFA $(\mathcal{A}_\varphi^D)^c$ od $\mathcal{A}$. Ideja je da se istovremeno konstruišu stablo prelaza od $\mathcal{A}_\varphi^D$ i graf prelaza od $(\mathcal{A}_\varphi^D)^c$. Ukoliko je $|X| = m$, tada su čvorovi traženog stabla i grafa $(2m+1)$-torke. Zajedno sa pokazivačima $s(\cdot)$ korišćenih u Algoritmu 6, koristimo i pokazivače $t(\cdot)$ koji prudružuju odgovarajuće cele brojeve čvorovima grafa prelaza koji konstruišemo. Procedura za konstrukciju grafa prelaza od $(\mathcal{A}_\varphi^D)^c$ data je Algoritmom 7.

Prema Teoremi 4.22, $(\mathcal{A}_\varphi^D)^c$ je konačan akko je $\mathcal{A}_\varphi^D$, i kao što je pokazano u Sekciji 4.5, $\mathcal{A}_\varphi^D$ je konačan akko $\mathcal{A}_\varphi$ zadovoljava slabo svojstvo reprezentativnih ciklusa, pri čemu je D maksimalna faktorizacija. Kao i u analizi Algoritma 6, posmatrajmo slučaj kada kolekcija $\{\varphi_u^D\}_{u \in X^*}$ ima najviše k^n različitih elemenata.

Primetimo da, kada se dostigne korak 19 u Algoritmu 7, tj. kada se konstruiše čvor u grafu G koji je $(2m+1)$-torka, prvih $2m$ komponenti je već izračunato u toku konstrukcije stabla prelaza T, i sve što preostaje je da se ukaže na te vrednosti (što se čini u $\mathcal{O}(2m)$ vremenu) i da se izračuna vrednost $\varphi_u^D \circ \tau$ (što se čini u $\mathcal{O}(n)$ vremenu).

Algoritam 7 Konstrukcija grafa prelaza kompletnog determinističkog dečjeg fazi automata $(\mathcal{A}_\varphi^D)^c$

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad kompletnom reziduiranom mrežom $\mathcal{L}$ i alfabetom $X = \{x_1, x_2, \ldots, x_m\}$, faktorizacija $D = (f, g)$ od A i fazi relacija φ na A.

Izlaz: KDDFA $(\mathcal{A}_\varphi^D)^c = ((A_\varphi^D)^c, (\sigma_\varphi^D)^c, (\delta_\varphi^D)^c, (\tau_\varphi^D)^c)$.

$\mathbf{S} \leftarrow 1, \mathbf{T} \leftarrow 1$
Inicijalizovati prazan red q čiji su elementi fazi podskupovi, prazno stablo T čiji su čvorovi takođe fazi podskupovi, prazan red w čiji su elementi $(2m+1)$-torke, i prazan graf G čiji su čvorovi takođe $(2m+1)$-torke.
$\varphi_\varepsilon^D \leftarrow f(\sigma \circ \varphi)$
$\texttt{Insert}(T, \varphi_\varepsilon^D)$
$s(\varphi_\varepsilon^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
$\texttt{Enqueue}(q, \varphi_\varepsilon^D)$
while not $\texttt{IsEmpty}(q)$ **do**
    $\varphi_u^D \leftarrow \texttt{Dequeue}(q)$
    for all $x \in X$ **do**
        $\varphi_{u\cdot x}^D \leftarrow f(\varphi_u^D \circ \delta_x \circ \varphi)$
        Zapamtiti vrednost $g(\varphi_u^D \circ \delta_x \circ \varphi)$
        if not $\texttt{Lookup}(T, \varphi_{u\cdot x}^D)$ **then**
            $\texttt{Enqueue}(q, \varphi_{u\cdot x}^D)$
            $s(\varphi_{u\cdot x}^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
        else
            $s(\varphi_{u\cdot x}^D) \leftarrow s(\texttt{GetValue}(T, \varphi_{u\cdot x}^D))$
        end if
        $\texttt{Insert}(T, \varphi_{u\cdot x}^D, \varphi_u^D, x/g(\varphi_u^D \circ \delta_x \circ \varphi))$
    end for
    $(\varphi_u^D)^c \leftarrow (\varphi_{ux_1}^D, g(\varphi_u^D \circ \delta_{x_1} \circ \varphi), \ldots, \varphi_{ux_m}^D, g(\varphi_u^D \circ \delta_{x_m} \circ \varphi), \varphi_u^D \circ \tau)$
    if $u = \varepsilon$ **then**
        $(\sigma_\varphi^D)^c((\varphi_u^D)^c) \leftarrow g(\sigma \circ \varphi)$
    else
        $(\sigma_\varphi^D)^c((\varphi_u^D)^c) \leftarrow 0$
    end if
    $(\tau_\varphi^D)^c((\varphi_u^D)^c) \leftarrow \varphi_u^D \circ \tau$
    if not $\texttt{Lookup}(G, (\varphi_u^D)^c)$ **then**
        $\texttt{Enqueue}(w, (\varphi_u^D)^c)$
        $t((\varphi_u^D)^c) \leftarrow \mathbf{T}, \quad \mathbf{T} \leftarrow \mathbf{T} + 1$
    else
        $t((\varphi_u^D)^c) \leftarrow t(\texttt{GetValue}(G, (\varphi_u^D)^c))$
    end if
    $\texttt{AddVertex}(G, (\varphi_u^D)^c)$
end while
while not $\texttt{IsEmpty}(w)$ **do**
    $(\varphi_u^D)^c \leftarrow \texttt{Dequeue}(w)$
    for all $x \in X$ **do**
        if $s(\varphi_{u\cdot x}^D) = s(\varphi_v^D)$ za neko $\varphi_v^D \in T$ sa jedinstvenom vrednošću $s(\varphi_v^D)$ **then**
            $\texttt{AddEdge}(G, (\varphi_u^D)^c, (\varphi_v^D)^c, x/g(\varphi_u^D \circ \delta_x \circ \varphi))$
        end if

41: **end for**
42: **end while**
43: Spajamo čvorove u grafu G sa istom vrednošću pokazivača t. Rezultujući dijagram predstavlja graf prelaza od $(\mathcal{A}_\varphi^D)^c$.

Dakle, korak 19 izvršava se u $\mathcal{O}(2m+n)$ vremenu. Korak 21 zahteva izvršenje u $\mathcal{O}(n^2(c_\otimes + c_\vee))$ vremenu. I pored toga, ukupno vreme izvršenja za korake 8–25 je $\mathcal{O}(mnk^n)$, kao što je izračunato u analizi Algoritma 6.

Fokusirajmo se sada na korak 26. U toku konstrukcije stabla prelaza T takođe smo odredili koji čvorovi u T su jednaki (u smislu jednakosti fazi skupova), te se njima pridružuju iste vrednosti pokazivača. Stoga, kada se proverava jednakost dve $(2m+1)$-torke u G, potrebno je jedino proveriti jednakost vrednosti pokazivača pridruženih čvorovima, a vreme izvršenja te provere je $\mathcal{O}(m)$. S obzirom da ne postoji više od k^n čvorova u grafu G, vreme izvršenja koraka 26 je $\mathcal{O}(mk^n)$. Ista vremenska složenost važi i za korak 30. Ukupno vreme izvršenje koraka 26–32 je $\mathcal{O}(mk^n)$.

Zaključujemo da je vreme izvršenja koraka 8–32 i dalje $\mathcal{O}(mnk^n)$, dok je vreme izvršenja koraka 7–33 jednako $\mathcal{O}(mnk^{2n})$. Na sličan način zaključujemo da izvršenje koraka 34–42 ne zahteva više vremena od izvršenja koraka 7–33. Sumirajući, zaključujemo da je ukupno vreme izvršenja Algoritma 7 jednako $\mathcal{O}(mnk^{2n})$, isto kao i za Algoritam 6.

4.5 Potrebni i dovoljni uslovi za determinizaciju preko maksimalne faktorizacije fazi podskupova i desno invarijantnih fazi kvazi-uređenja

U ovoj sekciji određujemo potrebne i dovoljne uslove da se Algoritmi iz Sekcije 4.4 završe u konačnom broju koraka. Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ fazi automat nad $\mathcal{L}$ i X, $D = (f, g)$ maksimalna faktorizacija od A, i φ desno invarijantno fazi kvazi-uređenje na A. S obzirom da je φ desno invarijantno fazi kvazi-uređenje na A, iz (4.24) dobijamo da za svaku reč $u = x_1 x_2 \dots x_n \in X^*$ važi:

$$\sigma_u \circ \varphi = \sigma \circ \delta_u \circ \varphi = \sigma \circ \varphi \circ \left(\bigcirc_{i=1}^{n} \delta_{x_i} \circ \varphi \right) = c_u^{\varphi,D} \otimes \varphi_u^D.$$

Štaviše, važi (4.11) i (4.21) jer je D maksimalna faktorizacija, te za svako $u \in X^*$ važi:

$$\varphi_u^D = f(\varphi_u^D) = f(c_u^{\varphi,D} \otimes \varphi_u^D),$$

te konačno zaključujemo da za svako $u \in X^*$ važi:

$$\varphi_u^D = f(\sigma_u \circ \varphi). \tag{4.27}$$

Sledeća Lema i Definicija dati su u radu de Mendívil i Garitagoitia [144].

Lema 4.23. *Neka je $\mathcal{A}$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, i $\mathcal{A}^D$ KDFA od $\mathcal{A}$. Tada su sledeći uslovi ekvivalentni:*

(a) *Postoji ceo broj $k > 0$ takav da za svaku reč $u \in X^*$, gde je $|u| \geq k$ i $\sigma_u \neq \emptyset$, postoje reči $v \in X^*$ i $w \in X^+$ gde je vw prefiks od u, tako da važi $f(\sigma_v) = f(\sigma_{vw})$.*

(b) *$\mathcal{A}^D$ je KDFKA.*

Dokaz. (a) $\Rightarrow$ (b). Pretpostavimo da tvrđenje (a) važi i da je A^D beskonačan skup. S obzirom da su elementi skupa A^D čvorovi stabla prelaza T konstruisanog preko Algoritma 6, postoji reč $q \in X^+$ za koju postoji nezatvorena grana, odnosno postoji beskonačna putanja $\pi_q \in P_q$ u stablu prelaza T od $\mathcal{A}^D$. Za svaki prefiks $q' \in X^*$ reči q važi $\sigma^D_{q'} \neq \emptyset$, jer bi se inače putanja π_q pod reči q završila u stanju $\sigma^D_{q'}$, s obzirom da bi tada važilo $\emptyset \circ \delta_x = \emptyset$ za svako $x \in X$, kao i $f(\emptyset) = \emptyset$. Štaviše, $\sigma^D_{q'} \neq \emptyset$ prema (4.5) i (4.27) povlači $\sigma_{q'} \neq \emptyset$, za svaki prefiks $q' \in X^*$ od q, s obzirom da je $\mathcal{L}$ bez delioca nule.

Sada, neka je $u = x_1x_2 \dots x_n \in X^*$ prefiks reči q, pri čemu je $n \geq k > 0$. Na osnovu prethodnog, važi i $\sigma_u \neq \emptyset$. Iz uslova (a) sledi da postoji prefiks vw od u tako da je $|w| > 0$ i $f(\sigma_v) = f(\sigma_{vw})$. Iz (4.27) zapravo imamo da važi $\sigma^D_v = \sigma^D_{vw}$. Stoga, putanja u stablu prelaza T koja prolazi kroz čvorove σ^D_ε, $\sigma^D_{x_1}$, …, σ^D_v, … završava se kada se dostigne čvor σ^D_v, što je kontradikcija sa početnom pretpostavkom.

(b) $\Rightarrow$ (a). Pretpostavimo da tvrđenje (b) važi, odnosno da je A^D konačan skup. Neka je $|A^D| = p$, i neka je $u = x_1x_2 \dots x_n \in X^*$ reč takva da je $n \geq p$ i $\sigma_u \neq \emptyset$. Tada je $\sigma_v \neq \emptyset$ za svaki prefiks v od u. Prema (4.5), svi elementi u nizu $f(\sigma)$, $f(\sigma_{x_1})$, …, $f(\sigma_{x_1x_2\dots x_n})$ različiti su od $\emptyset$. Štaviše, na osnovu (4.27) imamo da su elementi ovog niza elementi skupa A^D. Tada, prema Dirihleovom principu, postoje celi brojevi k i k' takvi da je $0 \leq k < k'$ i $f(\sigma_{x_1x_2\dots x_k}) = f(\sigma_{x_1x_2\dots x_{k'}})$. Neka je $v = x_1x_2 \dots x_k$ i $w = x_{k+1}x_{k+2} \dots x_{k'}$. Tada je vw prefiks od u za koji je $|w| > 0$ i $f(\sigma_v) = f(\sigma_{vw})$. Drugim rečima, tvrđenje (a) važi. □

Definicija 4.24 (**svojstvo reprezentativnih ciklusa**)**.** Neka je $\mathcal{A}$ FKA i $D = (f, g)$ maksimalna faktorizacija od A. Kažemo da $\mathcal{A}$ zadovoljava *svojstvo reprezentativnih ciklusa* ako važi:

SRC: Za sve reči $v \in X^*$ i $w \in X^+$, ako je $\sigma_{vw^m} \neq \emptyset$, za svaki ceo broj $m \geq 0$, tada postoji ceo broj $k \geq 0$ takav da za svaki ceo broj $r \geq k$ važi $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$.

Takođe, de Mendívil i Garitagoitia pokazali su u [144] da je SRC potreban i dovoljan uslov da $\mathcal{A}^D$ bude konačan, pod uslovom da je D maksimalna faktorizacija. Sada uvodimo širi koncept, takozvano *slabo svojstvo reprezentativnih ciklusa*, i pokazujemo da je ono potreban i dovoljan uslov da fazi automat $\mathcal{A}^D$ ima konačan broj stanja, pri čemu je D maksimalna faktorizacija. Na ovaj način, proširujemo klasu fazi automata koji mogu biti determinizovani preko maksimalne faktorizacije.

Definicija 4.25 (**slabo svojstvo reprezentativnih ciklusa**)**.** Neka je $\mathcal{A}$ FKA i $D = (f, g)$ maksimalna faktorizacija od A. Kažemo da $\mathcal{A}$ zadovoljava *slabo svojstvo reprezentativnih ciklusa* ako važi:

SSRC: Za sve reči $v \in X^*$ i $w \in X^+$, ako je $\sigma_{vw^m} \neq \emptyset$, za svaki ceo broj $m \geq 0$, tada postoje celi brojevi $k \geq 0$ i $r > k$ tako da važi $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$.

Za date dve reči $v \in X^*$ i $w \in X^+$, uslov "$\sigma_{vw^m} \neq \emptyset$, za svaki ceo broj $m \geq 0$" znači da postoji barem jedno stanje $a \in A$ u fazi automatu $\mathcal{A}$ tako da je $\sigma_{vw}(a) > 0$ (odnosno, stanje a koje je dostižno iz nekog početnog stanja fazi automata $\mathcal{A}$ preko reči vw) i $\delta_w(a, a) > 0$ (odnosno, postoji barem jedan ciklus koji uključuje stanje a preko reči w, a pritom je $|w| > 0$). Sa druge strane, uslov $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$ za cele brojeve $r \geq k$ iz SRC znači da ciklus preko reči w u fazi automatu $\mathcal{A}$ ne stvara beskonačan broj različitih stanja u KDFA $\mathcal{A}^D$ od $\mathcal{A}$, gde je D maksimalna faktorizacija od A. Ovo je stoga što su stanja $f(\sigma_{vw^r})$ za različite vrednosti celog broja r jednaka stanju $f(\sigma_{vw^k})$ za dati celi broj k. Preko SSRC pokazujemo da je samo

jedno $r > k$ potrebno da bi se izbegla generisanje beskonačnog broja stanja u $\mathcal{A}^D$ koji se nalaze u ciklusu u grafu od $\mathcal{A}^D$ preko reči w.

Očigledno, ako $\mathcal{A}$ zadovoljava SRC, tada $\mathcal{A}$ zadovoljava i SSRC.

U narednoj Teoremi dokazujemo da je SSRC dovoljan uslov da KDFA $\mathcal{A}^D$ od $\mathcal{A}$ ima konačan broj stanja.

Teorema 4.26. *Neka je $\mathcal{A}$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, i $\mathcal{A}^D$ KDFA od $\mathcal{A}$. Ako $\mathcal{A}$ zadovoljava SSRC, tada je $\mathcal{A}^D$ KDFKA.*

Dokaz. Dokaz izvodimo kontradikcijom. Pretpostavimo da $\mathcal{A}$ zadovoljava SSRC i da skup A^D ima beskonačan broj stanja. Prema Lemi 4.23, ovo znači da uslov (a) iz Leme 4.23 nije zadovoljen, Drugim rečima, važi sledeće:

Tvrdnja: Za svaki ceo broj $m > 0$ postoji reč $u \in X^*$, pri čemu je $|u| \geq m$ i $\sigma_u \neq \emptyset$, tako da za svako $p, q \in X^*$ gde je $|q| > 0$ i pq prefiks od u važi $f(\sigma_p) \neq f(\sigma_{pq})$.

Sada, odaberimo dve reči $v \in X^*$ i $w \in X^+$ tako da je $\sigma_{vw^m} \neq \emptyset$ za svaki ceo broj $m \geq 0$. To znači da postoji ciklus u grafu prelaza od $\mathcal{A}$ preko reči w koji uključuje stanje a koje je dostižno preko reči vw. Prema prethodnoj Tvrdnji, možemo odabrati dovoljno veliko $m > 0$ tako da važe sledeća dva tvrđenja:

1. Postoji $u \in X^*$ tako da je $|u| \geq m$ i $\sigma_u \neq \emptyset$,
2. Reč u prolazi kroz neki ciklus u grafu prelaza od $\mathcal{A}$ preko reči w najmanje r puta.

Dakle, vw^r je prefiks reči u, i s obzirom da je $\sigma_u \neq \emptyset$, tada je $\sigma_{vw^r} \neq \emptyset$. Prema SSRC, $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$. Neka je $p' = vw^k$ i $q' = w^{r-k}$. Primetimo da je $|q'| > 0$ jer je $r > k$. Drugim rečima, $f(\sigma_{p'q'}) = f(\sigma_{p'})$ gde je $|q'| > 0$, što je kontradikcija sa gore navedenom Tvrdnjom. □

SSRC iz Definicije 4.25 uključuje pojam maksimalne faktorizacije od skupa stanja fazi automata $\mathcal{A}$. Sledećom Lemom dokazujemo da je moguće izraziti SSRC isključivo preko elemenata familije fazi dostižnih stanja. To znači da je SSRC svojstvo interne strukture grafa prelaza fazi automata $\mathcal{A}$.

Lema 4.27. *Neka je $\mathcal{A}$ FKA i $D = (f, g)$ maksimalna faktorizacija od A. Fiksirajmo reči $v \in X^*$ i $w \in X^+$ takve da je $\sigma_{vw^m} \neq \emptyset$ za svaki ceo broj $m \geq 0$. Tada su sledeća svojstva ekvivalentna:*

(a) *Postoje $k \in \mathbb{N}_0$ i $r \in \mathbb{N}$, pri čemu je $r > k$, tako da je $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$,*

(b) *Postoje $k \in \mathbb{N}_0$, $r \in \mathbb{N}$ gde je $r > k$, kao i $l \in \mathcal{L}$ gde je $l > 0$, tako da važi $\sigma_{vw^r} = l \otimes \sigma_{vw^k}$.*

Dokaz. (a) $\Rightarrow$ (b). Pretpostavimo da svojstvo (a) važi. Tada je $\sigma_{vw^r} = \sigma_{vw^k} \circ \delta_{w^{r-k}}$, te s obzirom da je D maksimalna faktorizacija od A, (4.10) važi, te dobijamo:

$$g(\sigma_{vw^r}) = g(\sigma_{vw^k}) \otimes g(f(\sigma_{vw^k}) \circ \delta_{w^{r-k}}) = g(\sigma_{vw^k}) \otimes g(\sigma^D_{vw^k} \circ \delta_{w^{r-k}}).$$

Neka je $l = g(\sigma^D_{vw^k} \circ \delta_{w^{r-k}})$. Tada važi:

$$\sigma_{vw^r} = g(\sigma_{vw^r}) \otimes f(\sigma_{vw^r}) = g(\sigma_{vw^r}) \otimes f(\sigma_{vw^k}) = l \otimes (g(\sigma_{vw^k}) \otimes f(\sigma_{vw^k})) = l \otimes \sigma_{vw^k}.$$

Prema (4.3) imamo da je $l > 0$, te stoga sledi svojstvo (b).

(b) $\Rightarrow$ (a). Ako važi (b), tada je $f(\sigma_{vw^r}) = f(l \otimes \sigma_{vw^k})$. Prema pretpostavci, $\sigma_{vw^m} \neq \emptyset$ za svako $m \geq 0$, stoga je $\sigma_{vw^m} \neq \emptyset$, i štaviše $l > 0$, stoga je $l \otimes \sigma_{vw^m} \neq \emptyset$, za svako $m \geq 0$. S obzirom da je D maksimalna faktorizacija od A, dobijamo $f(\sigma_{vw^r}) = f(\sigma_{vw^k})$, te svojstvo (a) sledi. □

Definicija 4.28 (**slabo svojstvo reprezentativnih ciklusa (SSRC)**, alternativna reprezentacija)**.** Neka je $\mathcal{A}$ fazi automat. Tada $\mathcal{A}$ zadovoljava *slabo svojstvo reprezentativnih ciklusa (SSRC)* ako važi:

SSRC, alt: Za sve reči $v \in X^*$ i $w \in X^+$, ako je $\sigma_{vw^m} \neq \emptyset$, za svaki ceo broj $m \geq 0$, tada postoje celi brojevi $k \geq 0$ i $r > k$, kao i konstanta $l \in \mathcal{L}$ tako da je $l > 0$ i $\sigma_{vw^r} = l \otimes \sigma_{vw^k}$.

Prema Lemi 4.27, Definicije 4.25 i 4.28 su ekvivalentne, ali je SSRC u Definiciji 4.28 izraženo nezavisno od pojma maksimalne faktorizacije, što znači da je zavisno jedino od strukture grafa prelaza fazi automata. Imamo u vidu da je Teorema 4.26 takođe validna i za Definiciju 4.28.

U narednoj Teoremi pokazujemo da je SSRC potreban uslov da KDFA $\mathcal{A}^D$ od $\mathcal{A}$ ima konačan broj stanja, pri čemu je D *proizvoljna* faktorizacija od A.

Teorema 4.29. *Neka je $\mathcal{A}$ FKA, $D = (f, g)$ faktorizacija od A, i $\mathcal{A}^D$ KDFKA od $\mathcal{A}$. Tada $\mathcal{A}$ zadovoljava SSRC.*

Dokaz. Ponovo, dokaz ide preko kontradikcije - pretpostavimo da je skup A^D konačan i da $\mathcal{A}$ ne zadovoljava SSRC. To znači da postoje dve reči $v \in X^*$ i $w \in X^+$ takve da je $\sigma_{vw^m} \neq \emptyset$ za svaki ceo broj $m \geq 0$, i za svako $k \in \mathbb{N}_0$, svako $r \in \mathbb{N}$ sa $r > k$, i svako $l \in \mathcal{L}$ sa $l > 0$ važi $\sigma_{vw^r} \neq l \otimes \sigma_{vw^k}$.

S obzirom da postoje $v \in X^*$ i $w \in X^+$ takve da je $\sigma_{vw^m} \neq \emptyset$ za svako $m \geq 0$, zaključujemo da postoji ciklus u grafu prelaza od $\mathcal{A}$ preko reči w a koji prelazi preko stanja a koje je dostižno preko reči vw. Prema pretpostavci, fazi automat $\mathcal{A}^D$ je dobro definisan KDFKA ekvivalentan sa $\mathcal{A}$. Prema determinističkoj strukturi $\mathcal{A}^D$, postoji jedinstvena putanja u grafu prelaza od $\mathcal{A}^D$ za svako $vw^m \in X^*$, $m \geq 0$. Štaviše, $\mathcal{A}^D$ je konačan fazi automat, te stoga postoji $k \in \mathbb{N}_0$ i $j \in \mathbb{N}$ tako da postoji stanje u $\mathcal{A}^D$ koje je dostižno preko reči vw^k (naime, to je stanje $\sigma^D_{vw^k}$), i iz tog stanja postoji ciklus preko reči w^j. Iz (4.24) dobijamo:

$$\sigma_{vw^k} = c^D_{vw^k} \otimes \sigma^D_{vw^k}.$$

Neka je $w = x_1 x_2 \dots x_n$. Tada prema definiciji $c^D_{vw^{k+j}}$ dobijamo:

$$c^D_{vw^{k+j}} = c^D_{vw^k} \otimes \bigotimes_{t=0}^{j-1} \bigotimes_{i=1}^{n} \delta^D(\sigma^D_{wv^{k+t}x_1\dots x_{i-1}}, x_i, \sigma^D_{wv^{k+t}x_1\dots x_i})$$

Neka je $l = \bigotimes_{t=0}^{j-1} \bigotimes_{i=1}^{n} \delta^D(\sigma^D_{wv^{k+t}x_1\dots x_{i-1}}, x_i, \sigma^D_{wv^{k+t}x_1\dots x_i}) \in \mathcal{L}$. Tada iz (4.24) i ciklusa preko reči w^j dobijamo:

$$\sigma_{vw^{k+j}} = c^D_{vw^{k+j}} \otimes \sigma^D_{vw^{k+j}} = c^D_{vw^{k+j}} \otimes \sigma^D_{vw^k} = c^D_{vw^k} \otimes l \otimes \sigma^D_{vw^k} = l \otimes \sigma_{vw^k}. \tag{4.28}$$

Štaviše, s obzirom da je $\sigma_{vw^m} \neq \emptyset$ za svako $m \geq 0$, kao i $j > 0$, dobijamo da je $l > 0$. Dakle, (4.28) je kontradikcija sa početnom pretpostavkom, što je i bilo potrebno dokazati. □

4.6 Primeri

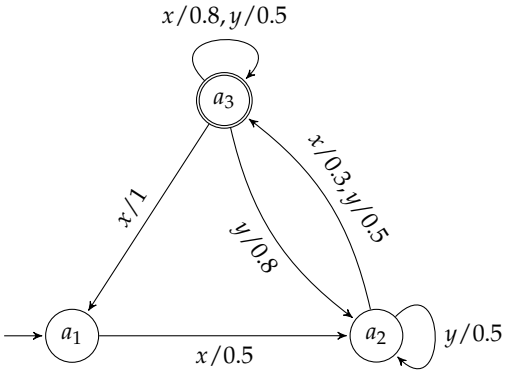
U ovoj sekciji dajemo nekoliko ilustrativnih primera kojima se porede performanse algoritama opisanih u prethodnim sekcijama. U svim primerima koristi se

Mohrieva faktorizacija $D = (f, g)$ definisana sa (4.13). Imamo u vidu da je Mohrieva faktorizacija maksimalna kada je $\mathcal{L}$ standardna proizvod algebra.

Napomena 4.30. U svim grafovima prelaza u narednim primerima ne prikazujemo prazno fazi stanje $\emptyset$.

Primer 4.31. Neka je $\mathcal{A}$ nad alfabetom $X = \{x, y\}$ i standardnom proizvod (Gougenovom) strukturom, čiji je graf prelaza prikazan na Slici 4.1. Fazi automat $\mathcal{A}$ može se prikazati i kao $\mathcal{A} = (A, \sigma, \delta, \tau)$, pri čemu je $A = \{a_1, a_2, a_3\}$, $\sigma = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, $\tau = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{-1}$, dok su matrice prelaza date sa:

$$\delta_x = \begin{bmatrix} 0 & 0.5 & 0 \\ 0 & 0 & 0.3 \\ 1 & 0 & 0.8 \end{bmatrix}, \quad \delta_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.8 & 0.5 \end{bmatrix}.$$



SLIKA 4.1: Graf prelaza za FKA $\mathcal{A}$ iz Primera 4.31

Fazi jezik $[\![\mathcal{A}]\!]$ ima beskonačni rang s obzirom da FKA $\mathcal{A}$, na primer, reči oblika x^2y^r prihvata u stepenu $[\![\mathcal{A}]\!](x^2y^r) = 0.15 \cdot (0.5)^r$, za svako $r \geq 0$. Dakle, Nerodov fazi automat $\mathcal{A}^N$ od $\mathcal{A}$ je beskonačan.

Sa druge strane, skup $A^D = \{\sigma_u^D\}_{u \in X^*}$ je konačan, što nije lako proveriti jer postoji više ciklusa u grafu prelaza od $\mathcal{A}$. Posmatrajmo, na primer, reči oblika x^*. Vidimo da je $\sigma_\varepsilon = [1\ 0\ 0]$, $\sigma_x = [0\ 0.5\ 0]$, $\sigma_{x^2} = [0\ 0\ 0.15]$, $\sigma_{x^3} = [0.15\ 0\ 0.12]$, $\sigma_{x^4} = [0.12\ 0.075\ 0.096]$, i $\sigma_{x^r} = 1.25^{r-4} \otimes \sigma_{x^4}$ za svako $r \geq 4$. Sumirajući, $\sigma_{\varepsilon x^m} \neq \emptyset$ za svako $m \geq 0$ i $\sigma_{\varepsilon x^r} = 1.25^{r-4} \otimes \sigma_{\varepsilon x^4}$ za $r \geq 4$, dakle svojstvo reprezentativnih ciklusa je zadovoljeno za reči ε and x.

Međutim, posmatrajmo reči oblika x^2y^m. U grafu prelaza od $\mathcal{A}$, postoji stanje koje je dostižno preko reči x^2y kao i ciklus preko tog stanja i reči y. U grafu prelaza od $\mathcal{A}^D$, postoji stanje dostižno preko reči x^2y, kao i ciklus preko tog stanja i reči y^2. Računom dobijamo da je $\sigma_{x^2y} = [0\ 0.12\ 0.75]$ i za svako $m \geq 3$:

$$\sigma_{x^2y^m} = \begin{cases} 0.4^{k-1} \otimes \begin{bmatrix} 0 & 0.06 & 0.06 \end{bmatrix}, & m = 2k. \\ 0.4^k \otimes \begin{bmatrix} 0 & 0.12 & 0.075 \end{bmatrix} = 0.4^{m-1/2}\sigma_{x^2y}, & m = 2k+1 \end{cases}.$$

Dakle, $\mathcal{A}$ ne zadovoljava svojstvo reprezentativnih ciklusa, ali zadovoljava slabo svojstvo reprezentativnih ciklusa za reči x^2 i y. Za ostale cikluse možemo se uveriti da svojstvo reprezentativnih ciklusa može ili ne mora biti zadovoljeno, dok je slabo svojstvo reprezentativnih ciklusa uvek zadovoljeno.

Dakle, KDFA $\mathcal{A}^D = (A^D, \sigma^D, \delta^D, \tau^D)$ od $\mathcal{A}$ je konačan. Elementi konačnog skupa A^D prikazani su u prvoj koloni u Tabeli 4.1. Unutrašnje kolone odgovaraju simbolima alfabeta X, i svako polje tabele sa indeksom vrste $\sigma_u^D \in A^D$ i indeksom kolone $p \in X$ predstavlja izraz $g(\sigma_u^D \circ \delta_p) \otimes f(\sigma_u^D \circ \delta_p)$. Polja u poslednjoj koloniu predstavljaju izraz $\tau^D(\sigma_u^D)$ za svako $\sigma_u^D \in A^D$. Računanje fazi skupa σ^D je trivijalno, naime $\sigma^D(\sigma_\varepsilon^D) = 1$ i $\sigma^D(\alpha) = 0$ za svako $\alpha \in A^D \setminus \{\sigma_\varepsilon^D\}$.

$\mathcal{A}^D$	x	y	τ^D
$\sigma_\varepsilon^D = [1\ 0\ 0]$	$0.5 \otimes [0\ 1\ 0] = 0.5 \otimes \sigma_x^D$	$1 \otimes \emptyset = 1 \otimes \sigma_y^D$	0
$\sigma_x^D = [0\ 1\ 0]$	$0.3 \otimes [0\ 0\ 1] = 0.3 \otimes \sigma_{x^2}^D$	$0.5 \otimes [0\ 1\ 1] = 0.5 \otimes \sigma_{xy}^D$	0
$\sigma_y^D = \emptyset$	$1 \otimes \emptyset = 1 \otimes \sigma_y^D$	$1 \otimes \emptyset = 1 \otimes \sigma_y^D$	0
$\sigma_{x^2}^D = [0\ 0\ 1]$	$1 \otimes [1\ 0\ 0.8] = 1 \otimes \sigma_{x^3}^D$	$0.8 \otimes [0\ 1\ 0.625] = 0.8 \otimes \sigma_{x^2y}^D$	1
$\sigma_{xy}^D = [0\ 1\ 1]$	$1 \otimes [1\ 0\ 0.8] = 1 \otimes \sigma_{x^3}^D$	$0.8 \otimes [0\ 1\ 0.625] = 0.8 \otimes \sigma_{x^2y}^D$	1
$\sigma_{x^3}^D = [1\ 0\ 0.8]$	$0.8 \otimes [1\ 0.625\ 0.8] = 0.8 \otimes \sigma_{x^4}^D$	$0.64 \otimes [0\ 1\ 0.625] = 0.64 \otimes \sigma_{x^2y}^D$	0.8
$\sigma_{x^2y}^D = [0\ 1\ 0.625]$	$0.625 \otimes [1\ 0\ 0.8] = 0.625 \otimes \sigma_{x^3}^D$	$0.5 \otimes [0\ 1\ 1] = 0.5 \otimes \sigma_{xy}^D$	0.625
$\sigma_{x^4}^D = [1\ 0.625\ 0.8]$	$0.8 \otimes [1\ 0.625\ 0.8] = 0.8 \otimes \sigma_{x^4}^D$	$0.64 \otimes [0\ 1\ 0.625] = 0.64 \otimes \sigma_{x^2y}^D$	0.8

TABELA 4.1: Vrednosti $g(\sigma_u^D \circ \delta_x) \otimes f(\sigma_u^D \circ \delta_x)$, $g(\sigma_u^D \circ \delta_y) \otimes f(\sigma_u^D \circ \delta_y)$ i $\tau^D(\sigma_u^D)$ izračunate preko Mohrieve faktorizacije D za svako $\sigma_u^D \in A^D$.

Slično, KDDFA $(\mathcal{A}^D)^c = ((A^D)^c, (\sigma^D)^c, (\delta^D)^c, (\tau^D)^c)$ od $\mathcal{A}$ takođe je konačan. Elementi konačnog skupa $(A^D)^c$ prikazani su u prvoj koloni Tabele 4.2. Unutrašnje kolone odgovaraju simbolima alfabeta X, a svako polje sa indeksom vrste $(\sigma_u^D)^c \in (A^D)^c$ i indeksom kolone $p \in X$ predstavlja izraz "$(\delta^D)^c((\sigma_u^D)^c, \delta_{x_p}, (\sigma_{up}^D)^c)/(\sigma_{up}^D)^c$". Polja u poslednjoj koloni predstavljaju izraz $(\tau^D)^c((\sigma_u^D)^c)$ za svako $(\sigma_u^D)^c \in (A^D)^c$. Fazi skup $(\sigma^D)^c$ računa se naravno trivijalno, odnosno $(\sigma^D)^c((\sigma_\varepsilon^D)^c) = 1$ i $(\sigma^D)^c(\alpha^c) = 0$ za svako $\alpha \in A^D \setminus \{\sigma_\varepsilon^D\}$.

$(\mathcal{A}_\varphi^D)^c$	x	y	$(\tau_\varphi^D)^c$
$(\sigma_\varepsilon^D)^c = (\sigma_x^D, 0.5, \sigma_y^D, 1, 0)$	$0.5/(\sigma_x^D)^c$	$1/\emptyset$	0
$(\sigma_x^D)^c = (\sigma_{x^2}^D, 0.3, \sigma_{xy}^D, 0.5, 0)$	$0.3/(\sigma_{x^2}^D)^c$	$0.5/(\sigma_{xy}^D)^c = 0.5/(\sigma_{x^2}^D)^c$	0
$(\sigma_{xy}^D)^c = (\emptyset, 1, \emptyset, 1, 0)$	$1/(\sigma_{xy}^D)^c$	$1/(\sigma_{xy}^D)^c$	0
$(\sigma_{x^2}^D)^c = (\sigma_{x^3}^D, 1, \sigma_{x^2y}^D, 0.8, 1)$	$1/(\sigma_{x^3}^D)^c$	$0.8/(\sigma_{x^2y}^D)^c$	1
$(\sigma_{x^3}^D)^c = (\sigma_{x^4}^D, 0.8, \sigma_{x^3y}^D, 0.64, 0.8)$	$0.8/(\sigma_{x^4}^D)^c = 0.8/(\sigma_{x^3}^D)^c$	$0.64/(\sigma_{x^3y}^D)^c = 0.64/(\sigma_{x^2y}^D)^c$	0.8
$(\sigma_{x^2y}^D)^c = (\sigma_{x^2yx}^D, 0.625, \sigma_{x^2y^2}^D, 0.5, 0.625)$	$0.625/(\sigma_{x^2yx}^D)^c = 0.625/(\sigma_{x^3}^D)^c$	$0.5/(\sigma_{x^2y^2}^D)^c = 0.5/(\sigma_{x^2}^D)^c$	0.625

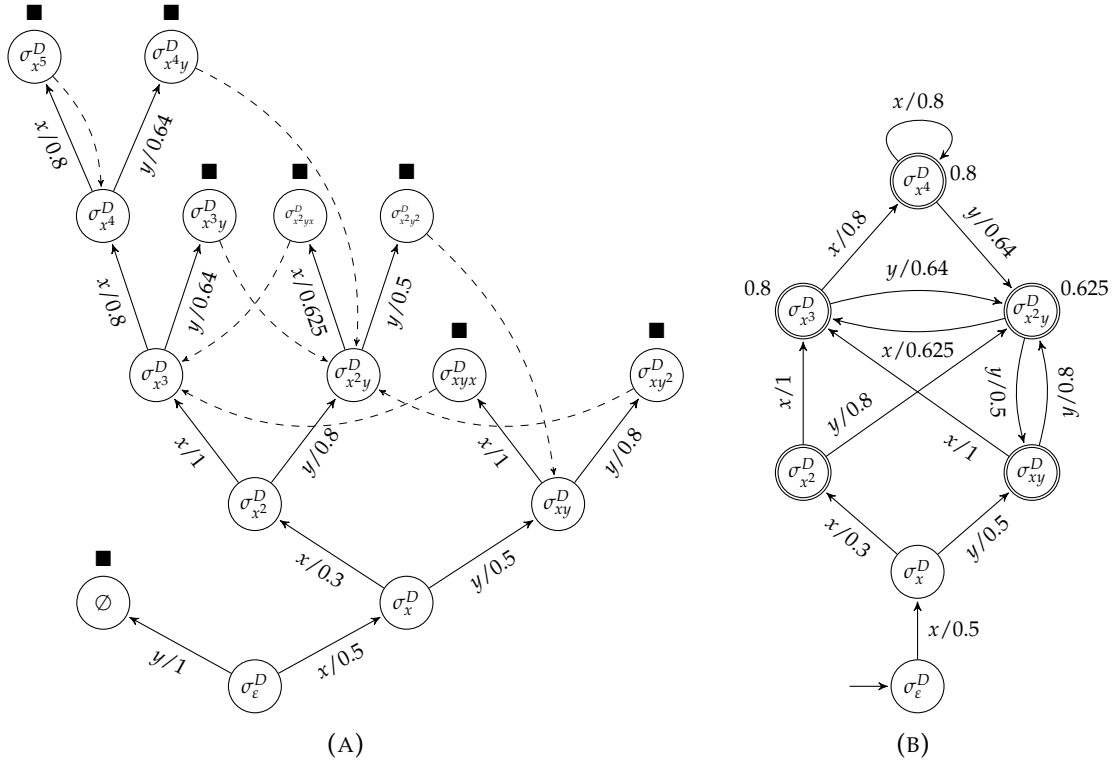
TABELA 4.2: Elementi i prelazi KDDFA $(\mathcal{A}^D)^c$ od $\mathcal{A}$ izračunate preko Mohrieve faktorizacije D.

Koristeći Algoritam 6 konstruišemo graf prelaza za KDFA $\mathcal{A}^D$ od $\mathcal{A}$ prikazanog na Slici 4.2, i koristeći Algoritam 7 konstruišemo graf prelaza za KDDFA $(\mathcal{A}^D)^c$ od $\mathcal{A}$ prikazanog na Slici 4.3. Odmah markiramo čvor $\sigma_y^D = \emptyset$ kao zatvoren u stablima prelaza u Figurama 4.2 i 4.3 radi pojednostavljenja. Primećujemo da KDFA ima 8 stanja (uključujući i stanje $\emptyset$), dok KDDFA ime 6 states, odnosno, KDDFA ima striktno manji broj stanja od odgovarajućeg KDFA.

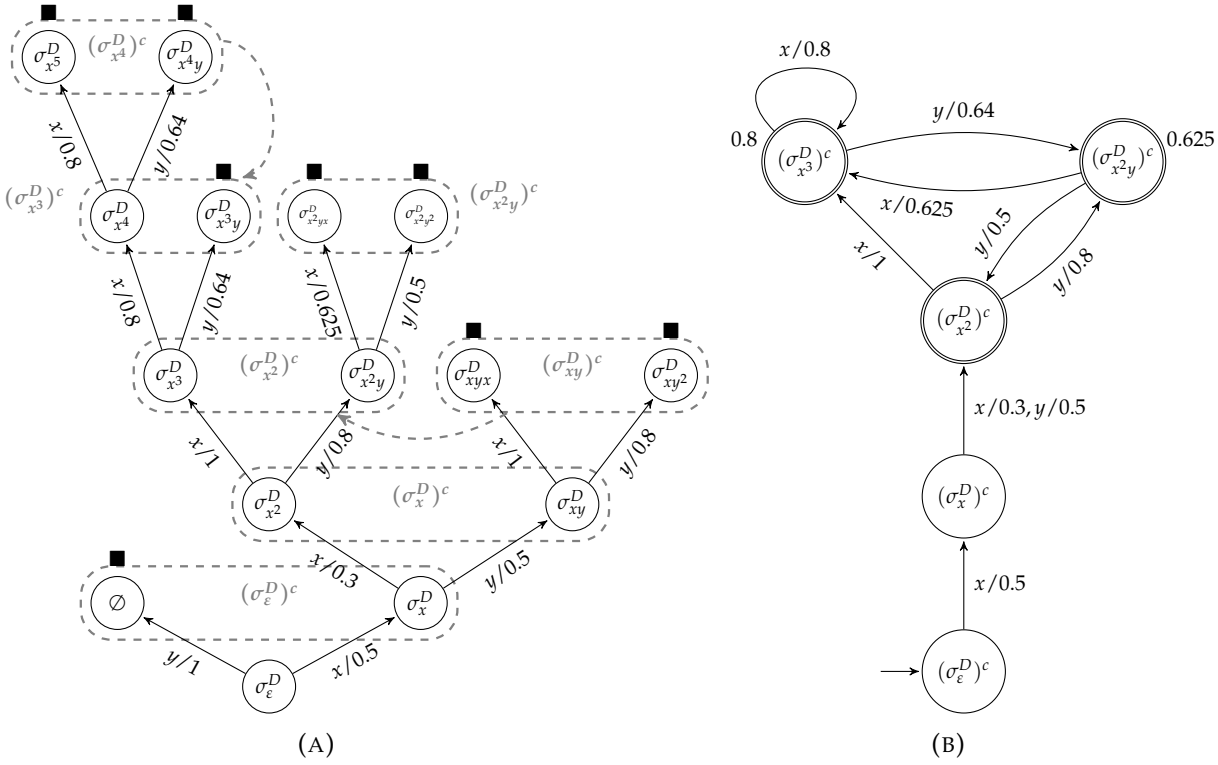
Štaviše, imamo da je najveće desno invarijantno fazi kvazi-uređenje na A jednako:

$$\varphi = \varphi^{\mathrm{ri}} = \begin{bmatrix} 1 & 0 & 0 \\ 0.6 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

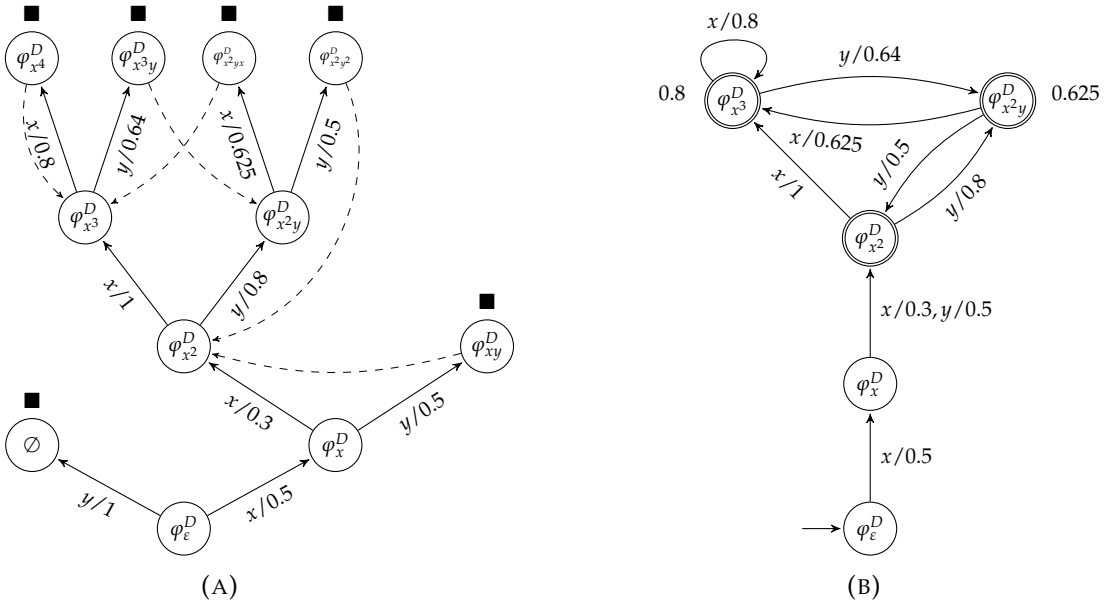
Fazi automat $\mathcal{A}_\varphi$ takođe zadovoljava slabo svojstvo reprezentativnih ciklusa, što se takođe može proveriti ručnom proverom. Graf prelaza od $\mathcal{A}_\varphi^D$ prikazan je na



SLIKA 4.2: Stablo prelaza (a) i graf prelaza (b) za KDFA $\mathcal{A}^D$ od fazi automata $\mathcal{A}$ iz Primera 4.31



SLIKA 4.3: Stablo prelaza (a) i graf prelaza (b) za KDDFA $(\mathcal{A}^D)^c$ od fazi automata $\mathcal{A}$ iz Primera 4.31



SLIKA 4.4: Stablo prelaza (a) i graf prelaza (b) za KDFA $\mathcal{A}_\varphi^D$ od fazi automata $\mathcal{A}$ iz Primera 4.31

Slici 4.4, a izračunat je primenom Algoritma 6. Elementi skupa A_φ^D u prvoj koloni Tabele 4.3. Unutrašnje kolone odgovaraju simbolima alfabeta X, a svako polje tabele sa indeksom vrste $\varphi_u^D \in A_\varphi^D$ i indeksom kolone $p \in X$ predstavlja izraz $g(\varphi_u^D \circ \delta_p \circ \varphi) \otimes f(\varphi_u^D \circ \delta_p \circ \varphi)$. Polja u poslednjoj koloni predstavljaju izraz $\tau_\varphi^D(\varphi_u^D)$ za svako $\varphi_u^D \in A_\varphi^D$.

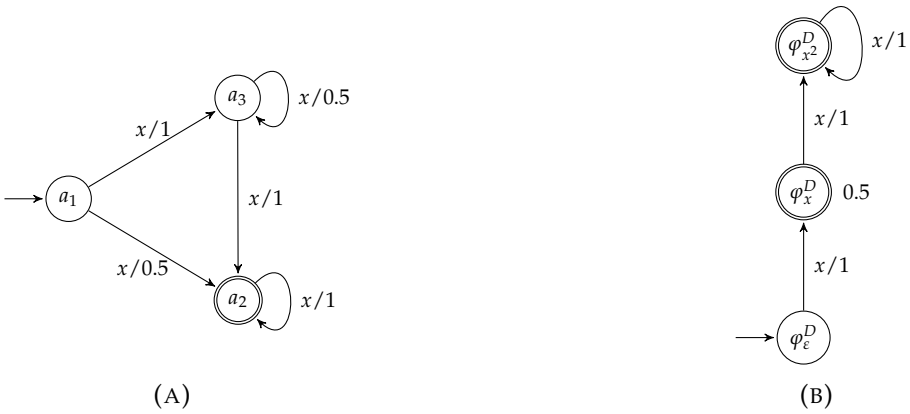
$\mathcal{A}_\varphi^D$	x	y	τ_φ^D
$\varphi_\varepsilon^D = [1\ 0\ 0]$	$0.5 \otimes [0.6\ 1\ 0] = 0.5 \otimes \varphi_x^D$	$1 \otimes \emptyset = 1 \otimes \varphi_y^D$	0
$\varphi_x^D = [0.6\ 1\ 0]$	$0.3 \otimes [1\ 1\ 1] = 0.3 \otimes \varphi_{x^2}^D$	$0.5 \otimes [1\ 1\ 1] = 0.5 \otimes \varphi_{x^2}^D$	0
$\varphi_y^D = \emptyset$	$1 \otimes \emptyset = 1 \otimes \varphi_y^D$	$1 \otimes \emptyset = 1 \otimes \varphi_y^D$	0
$\varphi_{x^2}^D = [1\ 1\ 1]$	$1 \otimes [1\ 0.8\ 0.8] = 1 \otimes \varphi_{x^3}^D$	$0.8 \otimes [0.625\ 1\ 0.625] = 0.8 \otimes \varphi_{x^2y}^D$	1
$\varphi_{x^3}^D = [1\ 0.8\ 0.8]$	$0.8 \otimes [1\ 0.8\ 0.8] = 0.8 \otimes \varphi_{x^3}^D$	$0.64 \otimes [0.625\ 1\ 0.625] = 0.64 \otimes \varphi_{x^2y}^D$	0.8
$\varphi_{x^2y}^D = [0.625\ 1\ 0.625]$	$0.625 \otimes [1\ 0.8\ 0.8] = 0.625 \otimes \varphi_{x^3}^D$	$0.5 \otimes [1\ 1\ 1] = 0.5 \otimes \varphi_{x^2}^D$	0.625

TABELA 4.3: Vrednosti $g(\varphi_u^D \circ \delta_x) \otimes f(\varphi_u^D \circ \delta_x)$, $g(\varphi_u^D \circ \delta_y) \otimes f(\varphi_u^D \circ \delta_y)$ i $\tau_\varphi^D(\varphi_u^D)$ izračunate preko Mohrieve faktorizacije D za svako $\varphi_u^D \in A_\varphi^D$.

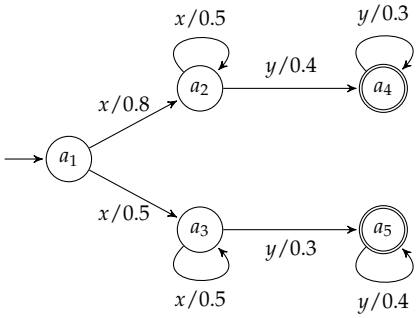
Naredni primer prikazuje pun potencijal našeg pristupa. Naime, dajemo primer fazi automata $\mathcal{A}$ čije je najveća desno invarijantno fazi kvazi-uređenje različito od identičke relacije, i pri tome $\mathcal{A}$ ne zadovoljava SSRC, dok $\mathcal{A}_\varphi$ zadovoljava, što čini fazi automat $\mathcal{A}$ determinizabilnim.

Primer 4.32. Neka je $\mathcal{A}$ FKA nad jednoelementnim alfabetom $X = \{x\}$ i standardnom proizvod algebrom čiji je graf prelaza prikazan na Slici 4.5 a).

Lako proveramao da je $\sigma_\varepsilon = [1\ 0\ 0]$, $\sigma_x = [0\ 0.5\ 1]$, kao i $\sigma_{x^n} = [0\ 1\ 0.5^{n-1}]$, za svako $n \in \mathbb{N}$, $n \geq 2$. Dakle, Nerodov fazi automat od $\mathcal{A}$ je beskonačan. Takođe, $\mathcal{A}$ ne zadovoljava SRC, jer nije moguće dobiti $l \in \mathcal{L}$ tako da je $l > 0$ i $\sigma_{x^* x^r} = l^{r-k} \otimes \sigma_{x^* x^k}$ za dato $k \geq 0$. Takođe, SSRC nije zadovoljeno. Dakle, KDFA $\mathcal{A}^D$ takođe je beskonačan.



SLIKA 4.5: Graf prelaza za fazi automat $\mathcal{A}$ iz Primera 4.32 (a), i KDFA $\mathcal{A}^D_\varphi$ od $\mathcal{A}$ (b).



SLIKA 4.6: Graf prelaza za fazi automat $\mathcal{A}$ iz Primera 4.33.

Sa druge strane, najveće desno invarijantno fazi kvazi uređenje, kao i najveće slabo desno invarijantno fazi kvazi-uređenje jednaki su:

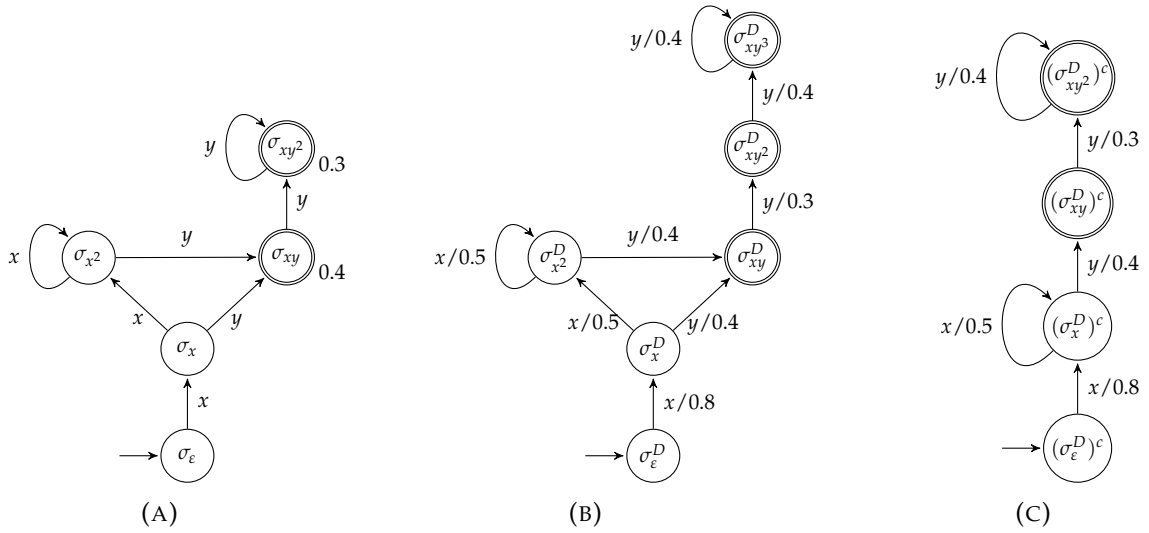
$$\varphi = \varphi^{\mathrm{ri}} = \varphi^{\mathrm{wri}} = \begin{bmatrix} 1 & 0 & 0.5 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

Jednostavno proveravamo da je $\sigma_\varepsilon \circ \varphi = [1\ 0\ 0.5]$, $\sigma_x \circ \varphi = [1\ 0.5\ 1]$, kao i $\sigma_{x^n} \circ \varphi = [1\ 1\ 1]$, za svako $n \in \mathbb{N}$, $n \geq 2$. Dakle, za $k = 2$ imamo da je $\sigma_{x^r} \circ \varphi = \sigma_{x^k} \circ \varphi$ za svaki ceo broj $r \geq k$, dakle $\mathcal{A}_\varphi$ zadovoljava SRC (time i SSRC). Dakle, Algoritam 6 završava se u konačnom broju koraka, i rezultujući KDFA $\mathcal{A}^D_\varphi$ od $\mathcal{A}$ prikazan je na Slici 4.5 b).

Naredni primer demonstrira determinizacione metode iz prethodnih sekcija na fazi automate koji prihvataju fazi jezik konačnog ranga.

Primer 4.33. Posmatrajmo fazi automat $\mathcal{A}$ nad alfabetom $X = \{x, y\}$ i standardnom Gödelovom algebrom, čiji je graf prelaza prikaza na Slici 4.6.

Lako proveravamo da fazi jezik raspoznat fazi automatom $\mathcal{A}$ ima konačan rang, s obzirom da je $\otimes$ idempotentna u Gödelovoj strukturi. Stoga je Nerodov fazi automat konačan, i prikazan je na Slici 4.7 a). KDFA od $\mathcal{A}$ prikazan je na Slici 4.7 b), dok je KDDFA od $\mathcal{A}$ prikazan na Slici 4.7 c).



SLIKA 4.7: Nerodov fazi automat (a), KDFA $\mathcal{A}^D$ od $\mathcal{A}$ (b), i KDDFA $(\mathcal{A}^D)^c$ od $\mathcal{A}$ (c), gde je $\mathcal{A}$ fazi automat iz Primera 4.33.

Glava 5

Kanonizacioni metod za fazi automate

Problem nalaženja minimalnog DKA ekvivalentnog početnom NKA-u predstavlja klasičan problem u računarskim naukama koji datira još iz radova Huffman [93] i Moore [156]. Iako su mnogi algoritmi za nalaženje minimalnog DKA razvijeni tokom godina, čini se da metod koji je razvio Brzozowski [29] privlači najviše pažnje istraživača zbog svoje jednostavnosti i elegantnosti. Naime, originalna ideja Brzozowskog [29] bila je da se, za dati ulazni NKA $\mathcal{A}$, konstruiše minimalni DKA $d(r(d(r(\mathcal{A}))))$ ekvivalentan sa $\mathcal{A}$, pri čemu je sa $r(\cdot)$ označen reverzni proces, dok je sa $d(\cdot)$ označen determinizacioni proces (na primer, dostižna podskupovna konstrukcija). Uprkos ekponencijalnoj složenosti u najgorem slučaju, algoritam Brzozowskog pokazao se dobrim u praktičnim primerima (videti [37, 200, 207]). Za više detalja o algoritmu Brzozowskog, kao i drugim algoritmima za nalaženje ekvivalentnog minimalnog DKA, videti [3, 4, 74, 75, 183].

Brojni radovi u skorijoj literaturi posvećeni su razvoju algoritama za nalaženje minimalnog determinističkog fazi automata za ulazni FKA. Jančić i Ćirić [108] bili su prvi koji su primenili proceduru Brzozowskog za nalaženje ekvivalentnog minimalnog k-DFA-a za dati FKA. Kasnije, Micić i drugi [147] razvili su kanonizacioni metod zasnovan na jezičkoj inkluziji, i iako je teoretski brži od metoda tipa Brzozowski [108], takođe su pokazali da se taj metod može koristiti u prvom reverznom procesu metoda tipa Brzozowski radi dodatnog ubrzanja.

U ovom radu, pod *kanonizacionom metodom* podrazumevamo determinizacioni metod koji za ulazni FKA daje ekvivalentni minimalni KDFA, a jedan takav metod najskorije je razvio de Mendívil [143]. U tom radu, autor je posmatrao konstrukciju tipa Brzozowski $d(r(d(r(\mathcal{A}))))$ za različite determinizacione procese $d(\cdot)$, za dati ulazni FKA $\mathcal{A}$. U istom radu dokazano je da ovakav metod rezultira minimalnim FDKA-om ukoliko determinizacioni proces $d(\cdot)$ ne proizvodi proporcionalne fazi skupove.

U ovoj glavi dajemo algoritam za kanonizaciju FKA tipa Brzozowski koja primenjuje determinizacione metode razvijene u prethodnoj Glavi. Ovaj metod zasniva se na korišćenju faktorizacije fazi podskupova, kao i levo invarijantnih fazi kvazi-uređenja. Konstrukcija tipa Brzozowski koja se zasniva na faktorizaciji fazi podskupova posmatrana je i u opštijem kontekstu u [143], ali mi ovde pokazujemo da se dodatna poboljšanja mogu postići paralelnom determinizacijom i sažimanjem ekvivalentnih stanja, koje se može postići primenom levo invarijantnih fazi kvazi-uređenja. Takođe, iako su se pojedini rezultati iz ove Glave pojavili u [143], mi ih i ovde navodimo jer se daje drugačiji dokaz.

Glava je organizovana na sledeći način. U Sekciji 5.1 uvodimo pojmove količnik fazi jezika i količnik fazi automata koji će biti neophodni za dokaz glavnih trvđenja. Sekcija 5.2 posvećena je formalnoj konstrukciji kanonizacionog metoda. Preciznije,

za dati FKA $\mathcal{A}$ čiji je skup stanja A, kao i fazi relaciju φ na A, najpre konstruišemo FKA $\tilde{\mathcal{A}}_\varphi$, a potom konstruišemo kompletan deterministički fazi automat $\tilde{\mathcal{A}}_\varphi^D$ od $\mathcal{A}$, gde je D faktorizacija od A. Na osnovu njega daje se način za konstrukciju fazi automata $\mathcal{B} = \widetilde{(\tilde{\mathcal{A}}_\varphi^D)}^{D_1}$ tipa Brzozowski, pri čemu je D_1 faktorizacija od $\mathcal{A}_\varphi^D$. Teoremom 5.10 obezbeđujemo da je fazi automat $\mathcal{B}$ minimalni KDFA ekvivalentan sa $\mathcal{A}$, pod uslovom da su faktorizacije D i D_1 maksimalne, a φ levo invarijantno fazi kvazi-uređenje. U Sekciji 5.3 formalno je predstavljen kanonizacioni algoritam zajedno sa vremenskom analizom. Na kraju, u Sekciji 5.4 prikazano je nekoliko ilustrativnih primera.

5.1 Količnik fazi jezika i količnik fazi automat

Kao i obično, neka je X neprazan alfabet i X^* slobodan monoid nad X. Tada definišemo *desni količnik*, ili jednostavno samo *količnik fazi jezika $h \in L^{X^*}$ od reči $u \in X^*$*, kao fazi jezi $u^{-1}h \in L^{X^*}$ sa:

$$u^{-1}h(v) = h(uv), \tag{5.1}$$

za svako $v \in X^*$. Lako se može proveriti da za svako $x \in X$ i $u \in X^*$ važi sledeća veza:

$$x^{-1}(u^{-1}h) = (ux)^{-1}h.$$

Neka je $h \in X^*$ fazi jezik i $D = (f, g)$ faktorizacija od X^*. Označimo sa $A_h^D = \{f(u^{-1}h)\}_{u \in X^*}$. Potom definišimo KDFA $\mathcal{A}_h^D = (A_h^D, \{a_0/\sigma_h^D(a_0)\}, \delta_h^D, \tau_h^D)$, pri čemu je A_h^D skup stanja fazi automata, $a_0 = f(\varepsilon^{-1}h)$ je početno stanje sa stepenom $\sigma_h^D(a_0) = g(\varepsilon^{-1}h)$, $\delta_h^D : A_h^D \times X \times A_h^D \to L$ je funkcija prelaza definisana sa:

$$\delta_h^D(\alpha, x, \beta) = \begin{cases} g(x^{-1}f(u^{-1}h)), & \alpha = f(u^{-1}h) \text{ i } \beta = f((ux)^{-1}h), \\ 0, & \text{inače,} \end{cases}$$

i $\tau_h^D : A_h^D \to L$ fazi skup završnih stanja definisan za svako $\alpha \in A_h^D$ sa:

$$\tau_h^D(\alpha) = \alpha(\varepsilon).$$

Tada važi sledeće tvrđenje:

Teorema 5.1. [142] *Neka je $h \in X^*$ fazi jezik i $D = (f, g)$ faktorizacija od X^*. Tada je fazi automat $\mathcal{A}_h^D$ minimalni KDFA koji raspoznaje fazi jezik h ako je $D = (f, g)$ maksimalna faktorizacija od X^*.*

Neka je $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ KDFA. Za svako stanje $q \in A$ definišimo novi KDFA sa $\mathcal{A}_q = (A, \{q/1\}, \delta, \tau)$, tj. fazi automat koji se dobija iz $\mathcal{A}$ zamenom početnog stanja a sa q sa stepenom 1, dok ostala stanja i prelazi ostaju nepromenjeni. Stoga iz (2.15) i (2.16) dobijamo da je:

$$[\![\mathcal{A}_q]\!](u) = \delta_*(q, u, q_u) \otimes \tau(q_u) = \tau_u(q), \tag{5.2}$$

za svako $u \in X^*$. Veza između fazi automata $[\![\mathcal{A}]\!]$ i $[\![\mathcal{A}_q]\!]$ je sledeća: za svako $u, v \in X^*$ imamo da je $[\![\mathcal{A}]\!](uv) = \sigma(a) \otimes \delta_*(a, u, a_u) \otimes [\![\mathcal{A}_{a_u}]\!](v)$. Stoga je:

$$u^{-1}[\![\mathcal{A}]\!] = (\sigma(a) \otimes \delta_*(a, u, a_u)) \otimes [\![\mathcal{A}_{a_u}]\!]. \tag{5.3}$$

za svako $u \in X^*$. Štaviše, lako se proverava da za svako $u \in X^*$ važi:

$$u^{-1}[\![\mathcal{A}_a]\!] = \delta_*(a, u, a_u) \otimes [\![\mathcal{A}_{a_u}]\!].$$

5.2 Konstrukcija reverznog kompletnog determinističkog fazi automata

Kao i u prethodnoj Glavi, u ovoj Glavi posmatramo kompletne reziduirane mreže bez delioca nule. Za dati FKA $\mathcal{A}$, najpre dajemo metod za računanje KDFA ekvivalentnog sa $\overline{\mathcal{A}}$.

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA i $D = (f, g)$ faktorizacija od A. Tada definišimo familiju fazi skupova $\{\tilde{\tau}_u^D\}_{u \in X^*}$ induktivno na sledeći način:

$$\begin{aligned} \tilde{\tau}_\varepsilon^D &= f(\tau), \\ \tilde{\tau}_{xu}^D &= f(\delta_x \circ \tilde{\tau}_u^D), \end{aligned}$$

za svako $u \in X^*$ and $x \in X$. Postavimo da je $\tilde{A}^D = \{\tilde{\tau}_u^D\}_{u \in X^*}$, i definišimo funkciju prelaza $\tilde{\delta}^D : \tilde{A}^D \times X \times \tilde{A}^D \to L$ sa:

$$\tilde{\delta}^D(\alpha, x, \beta) = \begin{cases} g(\delta_x \circ \tilde{\tau}_u^D), & \alpha = \tilde{\tau}_u^D \text{ i } \beta = \tilde{\tau}_{xu}^D, \quad u \in X^* \\ 0, & \text{inače,} \end{cases} \tag{5.4}$$

kao i fazi skup završnih stanja $\tilde{\tau}_\varphi^D : \tilde{A}_\varphi^D \to L$ sa:

$$\tilde{\tau}_\varphi^D(\alpha) = \sigma \circ \alpha, \quad \text{za svako } \alpha \in \tilde{A}_\varphi^D. \tag{5.5}$$

Tada važi sledeće tvrđenje:

Teorema 5.2. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA i $D = (f, g)$ faktorizacija od A. Tada je $\tilde{\mathcal{A}}^D = (\tilde{A}^D, \{\tilde{\tau}_\varepsilon^D / g(\tau)\}, \tilde{\delta}^D, \tilde{\tau}^D)$ dobro definisan KDFA ekvivalentan sa $\mathcal{A}$.*

Dokaz. Analogno dokazima Teorema 4.7 i 4.12. □

S obzirom da je $\tilde{\mathcal{A}}^D$ KDFA, proširujuće preslikavanje $(\tilde{\delta}^D)^* : \tilde{A}^D \times X^* \times \tilde{A}^D \to L$ preslikavanja $\tilde{\delta}^D : \tilde{A}^D \times X \times \tilde{A}^D \to L$, definisano preko (2.14), označavamo jednostavno sa $\tilde{\delta}^{D,*}$. Takođe, prema (2.15) imamo da je fazi jezik koji raspoznaje $\tilde{\mathcal{A}}^D$ jednak

$$[\![\tilde{\mathcal{A}}^D]\!](u) = \tilde{c}_u^D \otimes \tilde{\tau}^D(\tilde{\tau}_{\tilde{u}}^D), \tag{5.6}$$

za svako $u \in X^*$, pri čemu je $\tilde{c}_u^D \in \mathcal{L}$ definisano sa:

$$\tilde{c}_u^D = g(\tau) \otimes \tilde{\delta}^{D,*}(\tilde{\tau}_\varepsilon^D, u, \tilde{\tau}_{\tilde{u}}^D). \tag{5.7}$$

Lema 5.3. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA, $D = (f, g)$ faktorizacija od A, i neka je $\tilde{\mathcal{A}}^D = (\tilde{A}^D, \{\tilde{\tau}_\varepsilon^D / g(\tau)\}, \tilde{\delta}^D, \tilde{\tau}^D)$. Tada za svako $n \in \mathbb{N}_0$ and $x_1, x_2, \ldots, x_n \in X$ važi:*

$$\left(\bigcirc_{i=1}^{n} \delta_{x_{n+1-i}} \right) \circ \tau = \tilde{c}_{x_1 x_2 \ldots x_n}^D \otimes \tilde{\tau}_{x_n x_{n-1} \ldots x_1}^D, \tag{5.8}$$

gde je $\tilde{c}_{x_1 x_2 \ldots x_n}^D$ definisano sa (5.7).

Dalje, odaberimo proizvoljnu fazi relaciju $\varphi \in L^{A\times A}$ i konstruišimo fazi automat $\tilde{\mathcal{A}}_\varphi = (A, \tilde{\sigma}_\varphi, \tilde{\delta}_\varphi, \tilde{\tau}_\varphi)$ na sledeći način:

$$\begin{aligned}
&\tilde{\sigma}_\varphi = \sigma,\\
&\tilde{\delta}_\varphi(a,x,b) = (\varphi \circ \delta_x)(a,b), \quad \text{za svako } a,b \in A \text{ i } x \in X,\\
&\tilde{\tau}_\varphi = \varphi \circ \tau.
\end{aligned}$$

Označimo familiju familiju $(\tilde{A}_\varphi)^D = \{(\tilde{\tau}_\varphi)^D_u\}_{u\in X^*}$ sa $\tilde{A}^D_\varphi = \{\tilde{\varphi}^D_u\}_{u\in X^*}$, kao i odgovarajući fazi automat $(\tilde{\mathcal{A}}_\varphi)^D = ((\tilde{A}_\varphi)^D, \{(\tilde{\tau}_\varphi)^D_\varepsilon / g(\varphi\circ\tau)\}, (\tilde{\delta}_\varphi)^D, (\tilde{\tau}_\varphi)^D)$ jednostavno sa $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$. Naredni rezultati mogu biti dokazani na analogan način kao odgovarajući rezultati iz Glave 4, te je dokaz izostavljen.

Teorema 5.4. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA, $D = (f, g)$ faktorizacija od A, φ refleksivna slabo levo invarijantna fazi relacija na A, i neka je $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$. Tada je $[\![\overline{\mathcal{A}}]\!] = [\![\tilde{\mathcal{A}}^D_\varphi]\!]$.*

Lema 5.5. *Neka je $\mathcal{A}$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, φ fazi relacija na A, i neka je $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$. Tada za svako $\tilde{\varphi}^D_u \in \tilde{A}^D_\varphi$ with $u \in X^*$ važi: $f(\tilde{\varphi}^D_u) = \tilde{\varphi}^D_u$.*

Lema 5.6. *Neka je $\mathcal{A}$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, φ levo invarijantno fazi kvazi-uređenje na A, i neka je $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$. Tada za svako $u \in X^*$ važi:*

$$\tilde{\varphi}^D_u = f(\varphi \circ \tau_u). \tag{5.9}$$

Teorema 5.7. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, i neka su φ i ϕ levo invarijantna fazi kvazi-uređenja na A. Neka su $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$ i $\tilde{\mathcal{A}}^D_\phi = (\tilde{A}^D_\phi, \{\tilde{\phi}^D_\varepsilon / g(\phi\circ\tau)\}, \tilde{\delta}^D_\phi, \tilde{\tau}^D_\phi)$ odgovarajući KDFA-i ekvivalentni sa $\overline{\mathcal{A}}$. Ako je $\varphi \le \phi$, tada je $|\tilde{\mathcal{A}}^D_\phi| \le |\tilde{\mathcal{A}}^D_\varphi|$.*

Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA, $D = (f, g)$ maksimalna faktorizacija, i φ levo invarijantno fazi kvazi-uređenje na A. Kao što smo do sada videli, konstrukcijom fazi automata $\tilde{\mathcal{A}}^D_\varphi = (\tilde{A}^D_\varphi, \{\tilde{\varphi}^D_\varepsilon / g(\varphi\circ\tau)\}, \tilde{\delta}^D_\varphi, \tilde{\tau}^D_\varphi)$ dobijamo KDFA koji je ekvivalentan reverznom fazi automatu $\overline{\mathcal{A}}$ početnog fazi automata $\mathcal{A}$. Sada se postavlja pitanje - šta se dobija kada se isti postupak primeni na fazi automat $\tilde{\mathcal{A}}^D_\varphi$. Preciznije, ukoliko je $D_1 = (f_1, g_1)$ maksimalna faktorizacija od $\tilde{A}^D_\varphi$ i φ_1 levo invarijantno fazi kvazi-uređenje na $\tilde{A}^D_\varphi$, pitamo se da li je fazi automat $\widetilde{(\tilde{\mathcal{A}}^D_\varphi)}^{D_1}_{\varphi_1}$ KDFA ekvivalentan sa $\mathcal{A}$. U naredne dve Teoreme pokazujemo sledeće:

- Dobijeni fazi automat predstavlja minimalni KDFA ekvivalentan početnom fazi automatu,
- Fazi relacija φ_1 ne igra nikakvu ulogu u drugom reverznom procesu i može biti zamenjena jediničnom fazi relacijom.

Da bismo dokazali ove tvrđnje, koristimo činjenicu da drugi reverzni proces startuje od KDFA-a.

Lema 5.8. *Neka je $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ dostižni KDFA i $D = (f, g)$ maksimalna faktorizacija od X^*. Ako za svaki par stanja $p, q \in A$ pri čemu je $p \neq q$ važi $f([\![\mathcal{A}_p]\!]) \neq f([\![\mathcal{A}_q]\!])$, tada je $\mathcal{A}$ minimalan.*

Dokaz. S obzirom da je $D = (f, g)$ maksimalna faktorizacija od X^*, iz (5.3) dobijamo da je:

$$f([\![\mathcal{A}_{a_u}]\!]) = f(u^{-1}[\![\mathcal{A}]\!]).$$

Ukoliko označimo $h = [\![\mathcal{A}]\!]$, jasno je da je preslikavanje $\xi : A \to A_h^D$, definisano sa $\xi(p) = f([\![\mathcal{A}_p]\!])$, za svako $p \in A$, dobro definisano i surjektivno. Štaviše, ukoliko za svako $p, q \in A$ za koje je $p \neq q$ $f([\![\mathcal{A}_p]\!]) \neq f([\![\mathcal{A}_q]\!])$, tada preslikavanje ξ postaje i bijektivno. S obzirom da je fazi automat $\mathcal{A}_h^D$ minimalan (videti Teoremu 5.1), sledi da i fazi automat $\mathcal{A}$ mora biti minimalan. □

Teorema 5.9. *Neka je $\mathcal{A} = (A, \{a/\sigma(a)\}, \delta, \tau)$ dostižan KDFA, $D = (f, g)$ maksimalna faktorizacija od A, i neka je $\tilde{\mathcal{A}}^D = (\tilde{A}^D, \{\tilde{\tau}_\varepsilon^D / g(\tau)\}, \tilde{\delta}^D, \tilde{\tau}^D)$. Tada je $\tilde{\mathcal{A}}^D$ minimalni KDFA ekvivalentan sa $\overline{\mathcal{A}}$.*

Dokaz. Prema Lemi 5.2 i Teoremi 5.4, $\tilde{\mathcal{A}}^D$ ekvivalentan je sa $\overline{\mathcal{A}}$, te ostaje da dokažemo da je $\tilde{\mathcal{A}}^D$ minimalan. Pretpostavimo suprotno, odnosno da $\tilde{\mathcal{A}}^D$ nije minimalan. Prema Lemi 5.8 to znači da postoje reči $u, v \in X^*$ takve da je $\tilde{\tau}_u^D \neq \tilde{\tau}_v^D$ i $f'([\![\tilde{\mathcal{A}}_{\tilde{\tau}_u^D}^D]\!]) = f'([\![\tilde{\mathcal{A}}_{\tilde{\tau}_v^D}^D]\!])$, gde je $D' = (f', g')$ maksimalna faktorizacija od X^*. Prema (5.3) imamo da je:

$$\begin{aligned}
\bar{u}^{-1}[\![\tilde{\mathcal{A}}^D]\!] &= g(\tau) \otimes \tilde{\delta}_*^D(\tilde{\tau}_\varepsilon^D, \overline{u}, \tilde{\tau}_u^D) \otimes [\![\tilde{\mathcal{A}}_{\tilde{\tau}_u^D}^D]\!], \\
\bar{v}^{-1}[\![\tilde{\mathcal{A}}^D]\!] &= g(\tau) \otimes \tilde{\delta}_*^D(\tilde{\tau}_\varepsilon^D, \overline{v}, \tilde{\tau}_v^D) \otimes [\![\tilde{\mathcal{A}}_{\tilde{\tau}_v^D}^D]\!],
\end{aligned}$$

te s obzirom da je, prema pretpostavci, $f'([\![\tilde{\mathcal{A}}_{\tilde{\tau}_u^D}^D]\!]) = f'([\![\tilde{\mathcal{A}}_{\tilde{\tau}_v^D}^D]\!])$, za maksimalnu faktorizaciju $D' = (f', g')$ od X^*, dobijamo da je $f'(\bar{u}^{-1}[\![\tilde{\mathcal{A}}^D]\!]) = f'(\bar{v}^{-1}[\![\tilde{\mathcal{A}}^D]\!])$, ili ekvivalentno, $f'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) = f'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!])$. Stoga dobijamo:

$$\begin{aligned}
g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \bar{u}^{-1}[\![\overline{\mathcal{A}}]\!] &= g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes g'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes f'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \\
&= g'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes f'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \\
&= g'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]. \qquad (5.10)
\end{aligned}$$

Sa druge strane, za svako $w \in X^*$ važi:

$$\begin{aligned}
[\![\overline{\mathcal{A}}]\!](\bar{u}\bar{w}) = \overline{[\![\mathcal{A}]\!]}(\bar{u}\bar{w}) = [\![\mathcal{A}]\!](wu) &= \sigma(a) \otimes \delta_*(a, w, a_w) \otimes [\![\mathcal{A}_{a_w}]\!](u) \\
&= \sigma(a) \otimes \delta_*(a, w, a_w) \otimes \tau_u(a_w),
\end{aligned}$$

pri čemu poslednja jednakost sledi iz (5.2). Analogno je:

$$[\![\overline{\mathcal{A}}]\!](\bar{v}\bar{w}) = \sigma(a) \otimes \delta_*(a, w, a_w) \otimes \tau_v(a_w),$$

za svako $w \in X^*$. S obzirom da je $\sigma(a) \otimes \delta_*(a, w, a_w) > 0$ (na osnovu činjenice da je $\mathcal{A}$ kompletan i deterministički), kao i $g'(h) > 0$ za svako $h \in L^{X^*}$, takođe dobijamo da je:

$$\begin{aligned}
c_1 &= g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \sigma(a) \otimes \delta_*(a, w, a_w) > 0, \\
c_2 &= g'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \sigma(a) \otimes \delta_*(a, w, a_w) > 0,
\end{aligned}$$

jer je $\mathcal{L}$ bez delioca nule. Stoga je za svako $w \in X^*$:

$$(g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \bar{u}^{-1}[\![\overline{\mathcal{A}}]\!])(\bar{w}) = g'(\bar{v}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \bar{u}^{-1}[\![\overline{\mathcal{A}}]\!](\bar{w}) = c_1 \otimes \tau_u(a_w),$$

i analogno,

$$(g'(\bar{u}^{-1}[\![\overline{\mathcal{A}}]\!]) \otimes \bar{v}^{-1}[\![\overline{\mathcal{A}}]\!])(\bar{w}) = c_2 \otimes \tau_v(a_w),$$

za svako $w \in X^*$. Stoga, iz (5.10) sledi da je:

$$c_1 \otimes \tau_u = c_2 \otimes \tau_v.$$

S obzirom da je $c_1 > 0$ i $c_2 > 0$, imamo da je $f(\tau_u) = f(\tau_v)$. Ali, iz (5.9) dobijamo da je $f(\tau_u) = \tilde{\tau}_u^D$ i $f(\tau_v) = \tilde{\tau}_v^D$, što znači da je:

$$\tilde{\tau}_u^D = \tilde{\tau}_v^D.$$

Dakle, dobili smo kontradikciju sa početnom pretpostavkom, što je i trebalo dokazati. □

Sada smo u poziciji da dokažemo glavno tvrđenje ove Sekcije.

Teorema 5.10. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA, $D = (f, g)$ maksimalna faktorizacija od A, φ levo invarijantno fazi kvazi-uređenje na A, i $D_1 = (f_1, g_1)$ maksimalna faktorizacija od A_φ^D. Neka je fazi automat $\mathcal{B}$ definisan sa:*

$$\mathcal{B} = \widetilde{(\tilde{\mathcal{A}}_\varphi^D)}^{D_1}.$$

Tada je $\mathcal{B}$ minimalni KDFA ekvivalentan sa $\mathcal{A}$.

Dokaz. Prema Lemi 5.2 i Teoremi 5.4, $\tilde{\mathcal{A}}_\varphi^D$ je KDFA ekvivalentan sa $\overline{\mathcal{A}}$, i $\widetilde{(\tilde{\mathcal{A}}_\varphi^D)}^{D_1}$ KDFA ekvivalentan sa $\mathcal{A}$. Štaviše, prema Teoremi 5.9, $\mathcal{B}$ je minimalni KDFA ekvivalentan sa $\mathcal{A}$. □

5.3 Algoritam za kanonizaciju i njegova analiza

U ovoj Sekciji dajemo algoritam za računanje grafa prelaza KDFA konstruisanog u Teoremi 5.10. Koristimo sve strukture podataka i operacije nad njima koje su korišćene u Sekciji 4.4. Metod za računanje KDFA $\mathcal{B} = \widetilde{(\tilde{\mathcal{A}}_\varphi^D)}^{D_1}$ dat je sukcesivnom primenom Algoritama 8 i 9, pri čemu je prvi reverzni proces formalizovan Algoritmom 8, dok je drugi reverzni proces formalizovan Algoritmom 9. Primetimo da se drugi reverzni proces malo razlikuje od prvog, s obzirom da je ulaz u drugom reverznom procesu KDFA.

Uslovi pod kojima se Algoritam 8, te stoga i Algoritam 9 završava u konačnom broju koraka može se odrediti analogno kao za Algoritam 6 konstruisan u Glavi 4. Ovaj uslov nazivamo *reverzno slabo svojstvo reprezentativnih ciklusa*. Formalni dokaz da je ovaj uslov potreban i dovoljan uslov da bi se Algoritam 8 završio u konačnom broju koraka je izostavljen, s obzirom da je vrlo sličan onome dat u Glavi 4.

Definicija 5.11 (reverzno slabo svojstvo reprezentativnih ciklusa (RSSRC)). Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA i $D = (f, g)$ maksimalna faktorizacija od A. Kažemo da $\mathcal{A}$ zadovoljava *reverzno slabo svojstvo reprezentativnih ciklusa (RSSRC)* ako važi sledeće:

RSSRC: Za sve reči $v \in X^*$ and $w \in X^+$, ako je $\tau_{\bar{w}^m\bar{v}} \neq \emptyset$, za svaki ceo broj $m \geq 0$, tada postoje celi brojevi $k \geq 0$ i $r \geq k$ tako da je $f(\varphi \circ \tau_{\bar{w}^r\bar{v}}) = f(\varphi \circ \tau_{\bar{w}^k\bar{v}})$.

Algoritam 8 Konstrukcija grafa prelaza kompletnog determinističkog fazi automata $\tilde{\mathcal{A}}_\varphi^D$

Ulaz: Fazi automat $\mathcal{A} = (A, \sigma, \delta, \tau)$ nad kompletnom reziduiranom mrežom $\mathcal{L}$ i alfabetom X, faktorizacija $D = (f, g)$ od A, i fazi relacija φ na A.

Izlaz: Kompletan deterministički fazi automat $\tilde{\mathcal{A}}_\varphi^D = (\tilde{A}_\varphi^D, \{\tilde{\varphi}_\varepsilon^D / g(\varphi \circ \tau)\}, \tilde{\delta}_\varphi^D, \tilde{\tau}_\varphi^D)$.

1: $\mathbf{S} \leftarrow 1$
2: Inicijalizovati prazan red q čiji su elementi fazi skupovi, kao i prazno stablo T čiji su čvorovi takođe fazi skupovi
3: $\tilde{\varphi}_\varepsilon^D \leftarrow f(\varphi \circ \tau)$
4: $\texttt{Insert}(T, \tilde{\varphi}_\varepsilon^D)$
5: $\tilde{\tau}_\varphi^D(\tilde{\varphi}_\varepsilon^D) \leftarrow \sigma \circ \tilde{\varphi}_\varepsilon^D$
6: $s(\tilde{\varphi}_\varepsilon^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
7: $\texttt{Enqueue}(q, \tilde{\varphi}_\varepsilon^D)$
8: **while** not $\texttt{IsEmpty}(q)$ **do**
9: $\quad \tilde{\varphi}_u^D \leftarrow \texttt{Dequeue}(q)$
10: $\quad$ **for all** $x \in X$ **do**
11: $\quad\quad \tilde{\varphi}_{x\cdot u}^D \leftarrow f(\varphi \circ \delta_x \circ \tilde{\varphi}_u^D)$
12: $\quad\quad \tilde{\tau}_\varphi^D(\tilde{\varphi}_{x\cdot u}^D) \leftarrow \sigma \circ \tilde{\varphi}_{x\cdot u}^D$
13: $\quad\quad$ **if** not $\texttt{Lookup}(T, \tilde{\varphi}_{x\cdot u}^D)$ **then**
14: $\quad\quad\quad \texttt{Enqueue}(q, \tilde{\varphi}_{x\cdot u}^D)$
15: $\quad\quad\quad s(\tilde{\varphi}_{x\cdot u}^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
16: $\quad\quad$ **else**
17: $\quad\quad\quad s(\tilde{\varphi}_{x\cdot u}^D) \leftarrow s(\texttt{GetValue}(T, \tilde{\varphi}_{x\cdot u}^D))$
18: $\quad\quad$ **end if**
19: $\quad\quad \texttt{Insert}(T, \tilde{\varphi}_{x\cdot u}^D, \tilde{\varphi}_u^D, x/g(\varphi \circ \delta_x \circ \tilde{\varphi}_u^D))$
20: $\quad$ **end for**
21: **end while**
22: Stablo prelaza T je konstruisano. Spajamo listove stabla T sa njegovim unutrašnjim čvorovima sa istom vrednošću pokazivača s. Postavljamo čvorove rezultujućeg dijagrama kao elemente skupa $\tilde{A}_\varphi^D$. Definišemo funkciju fazi prelaza $\tilde{\delta}_\varphi^D : \tilde{A}_\varphi^D \times X \times \tilde{A}_\varphi^D \to L$ sa $\tilde{\delta}_\varphi^D(\alpha, x, \beta) = t$ ukoliko postoji grana označena sa x/t iz čvora sa vrednošću α u čvor sa vrednošću β u rezultujućem dijagramu, a $\tilde{\delta}_\varphi^D(\alpha, x, \beta) = 0$ u suprotnom. Rezultujući fazi automat označavamo sa $\tilde{\mathcal{A}}_\varphi^D = (\tilde{A}_\varphi^D, \{\tilde{\varphi}_\varepsilon^D / g(\varphi \circ \tau)\}, \tilde{\delta}_\varphi^D, \tilde{\tau}_\varphi^D)$.

Definicija 5.12 (**reverzno slabo svojstvo reprezentativnih ciklusa (RSSRC)**, alternativna reprezentacija)**.** Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ FKA . Kažemo da $\mathcal{A}$ zadovoljava *reverzno slabo svojstvo reprezentativnih ciklusa (RSSRC)* ako važi sledeće:

RSSRC, alt: Za sve reči $v \in X^*$ i $w \in X^+$, ako je $\tau_{\bar{w}^m \bar{v}} \neq \emptyset$, za svaki ceo broj $m \geq 0$, tada postoje celi brojevi $k \geq 0$ i $r \geq k$, kao i konstanta $l \in \mathcal{L}$ tako da je $l > 0$ i $\varphi \circ \tau_{\bar{w}^r \bar{v}} = l \otimes (\varphi \circ \tau_{\bar{w}^k \bar{v}})$.

Analizirajmo sada vreme izvršenja metoda koji se sastoji od sukcesivne primene Algoritama 8 i 9. Neka je $\mathcal{A}$ FKA nad $\mathcal{L}$ i X sa $|\mathcal{A}| = n$ i $|X| = m$, $D = (f, g)$ faktorizacija od A, $D_1 = (f_1, g_1)$ faktorizacija od A_φ^D, i φ fazi relacija na A. Neka $\tilde{\mathcal{A}}_\varphi$ zadovoljava RSSRC. U tom slučaju, označimo sa k broj različitih vrednosti koje konstruisani fazi skupovi mogu da uzmu. Tada od tih k različitih vrednosti možemo konstruisati najviše k^n fazi skupova na A, odnosno, familija $\{\tilde{\varphi}_u^D\}_{u \in X^*}$ ima najviše

Algoritam 9 Konstrukcija minimalnog kompletnog deterministčkog fazi automata

Ulaz: Kompletan deterministički fazi automat $\tilde{\mathcal{A}}_\varphi^D = (\tilde{A}_\varphi^D, \{\tilde{\varphi}_\varepsilon^D / g(\varphi \circ \tau)\}, \tilde{\delta}_\varphi^D, \tilde{\tau}_\varphi^D)$ nad kompletnom reziduiranom mrežom $\mathcal{L}$ i alfabetom X, i faktorizacija $D = (f, g)$ od $\tilde{A}_\varphi^D$.

Izlaz: Minimalani kompletan deterministički fazi automat $\mathcal{B} = (B, \{b_0 / \sigma^B(b_0)\}, \delta^B, \tau^B)$.

23: $\mathbf{S} \leftarrow 1$
24: Inicijalizovati prazan red q čiji su elementi fazi skupovi, kao i prazno stablo T čiji su čvorovi takođe fazi skupovi
25: $\tilde{\tau}_\varepsilon^D \leftarrow f(\tilde{\tau}_\varphi^D)$
26: $\texttt{Insert}(T, \tilde{\tau}_\varepsilon^D)$
27: $b_0 \leftarrow \tilde{\tau}_\varepsilon^D, \quad \sigma^B(b_0) \leftarrow g(\tilde{\tau}_\varphi^D)$
28: $\tau^B(\tilde{\tau}_\varepsilon^D) \leftarrow g(\varphi \circ \tau) \otimes \tilde{\tau}_\varepsilon^D(\tilde{\varphi}_\varepsilon^D)$
29: $s(\tilde{\tau}_\varepsilon^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
30: $\texttt{Enqueue}(q, \tilde{\tau}_\varepsilon^D)$
31: **while** not $\texttt{IsEmpty}(q)$ **do**
32: $\quad \tilde{\tau}_u^D \leftarrow \texttt{Dequeue}(q)$
33: $\quad$ **for all** $x \in X$ **do**
34: $\quad\quad \tilde{\tau}_{x \cdot u}^D \leftarrow f(\delta_x \circ \tilde{\varphi}_u^D)$
35: $\quad\quad \tilde{\tau}^B(\tilde{\tau}_{x \cdot u}^D) \leftarrow g(\varphi \circ \tau) \otimes \tilde{\tau}_{x \cdot u}^D(\tilde{\varphi}_\varepsilon^D)$
36: $\quad\quad$ **if** not $\texttt{Lookup}(T, \tilde{\tau}_{x \cdot u}^D)$ **then**
37: $\quad\quad\quad \texttt{Enqueue}(q, \tilde{\tau}_{x \cdot u}^D)$
38: $\quad\quad\quad s(\tilde{\tau}_{x \cdot u}^D) \leftarrow \mathbf{S}, \quad \mathbf{S} \leftarrow \mathbf{S} + 1$
39: $\quad\quad$ **else**
40: $\quad\quad\quad s(\tilde{\tau}_{x \cdot u}^D) \leftarrow s(\texttt{GetValue}(T, \tilde{\tau}_{x \cdot u}^D))$
41: $\quad\quad$ **end if**
42: $\quad\quad \texttt{Insert}(T, \tilde{\tau}_{x \cdot u}^D, \tilde{\tau}_u^D, x / g(\delta_x \circ \tilde{\tau}_u^D))$
43: $\quad$ **end for**
44: **end while**
45: Stablo prelaza T je konstruisano. Spajamo listove stabla T sa njegovim unutrašnjim čvorovima sa istom vrednošću pokazivača s. Rezultujući dijagram predstavlja graf prelaza od $\mathcal{B}$.

k^n različitih članova. Algoritam 8 analogan je Algoritmu 6 iz Glave 4, te je njegovo vreme izvršenja jednako $\mathcal{O}(mnk^{2n})$.

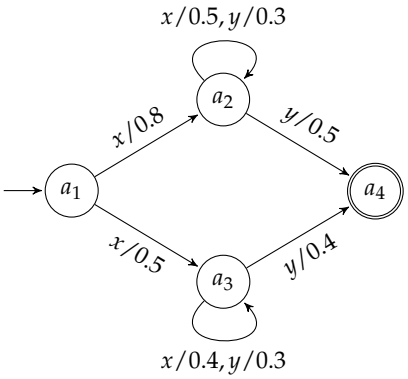
Iako Algoritam 4 može za ulaz imati eksponencijalno veliki fazi automat, njegov izlaz je minimalni KDFA koji nema veći broj stanja od $\mathcal{A}_\varphi^D$ konstruisanog preko Algoritma 6, a koji ne može imati više od k^n stanja. Dakle, rezultujuće stablo prelaza konstruisano u Algoritmu 9 ne može imati više od k^n unutrašnjih čvorova, te je stoga i ukupno vreme izvršenja Algoritma 9 takođe jednako $\mathcal{O}(mnk^{2n})$. Što znači da je ukupno vreme izvršenja konstrukcije minimalnog KDFA-a $\mathcal{B} = \widetilde{(\tilde{\mathcal{A}}_\varphi^D)}^{D_1}$ jednako $\mathcal{O}(mnk^{2n})$, odnosno, isto kao i za Algoritam 6.

5.4 Primeri

U ovoj Sekciji dajemo nekoliko primera koji ilustruju rad Algoritama 8 i 9, kao i poređenje sa Algoritmima iz Glave 4. Najpre dajemo primer u kojem prikazujemo

da je izlaz prvog reverznog procesa konačni fazi automat kada je najveće levo invarijantno fazi kvazi-uređenje na ulaznom fazi automatu različito od identičke fazi relacije.

Primer 5.13. Neka je $\mathcal{A}$ FKA nad alfabetom $X = \{x, y\}$ i standardnom proizvod algebrom, a čiji je graf prelaza prikazan na Slici 5.1.



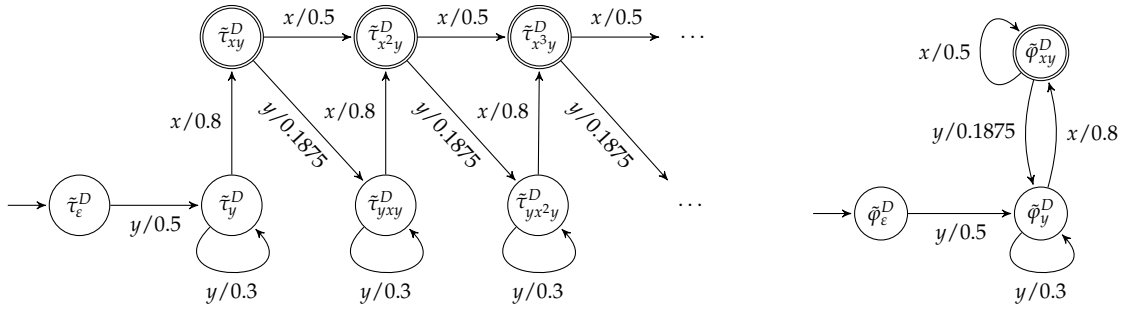
SLIKA 5.1: Graf prelaza za FKA $\mathcal{A}$ iz Primera 5.13

Neka je $D = (f, g)$ Mohireva faktorizacija od A, a $D_1 = (f_1, g_1)$ Mohrieva faktorizacija od A^D. Možemo se uveriti da RSSRC nije zadovoljen za fazi automat $\mathcal{A}$. Zaista, imamo da je $\tau_y = [0\ 0.5\ 0.4\ 0]$ i $\tau_{x^k y} = \left[0.4 \cdot (1/2)^{k-1}\ 0.5^{k+1}\ 0.4^{k+1}\ 0\right]$, tako da ne postoje skalar $l \in \mathcal{L}\backslash\{0\}$ i celi brojevi $k \geq 0$ i $r \geq k$ tako da je $\tau_{x^r y} = l \otimes \tau_{x^k y}$. Dakle, KDFA $\tilde{\mathcal{A}}^D$ (ekvivalentan sa $\overline{\mathcal{A}}$) je beskonačan, i deo njegovog grafa prelaza prikazan je na Slici 5.2 a). Odnosno, prvi reverzni proces ne završava se u konačnom broju koraka, i primena Algoritma nije moguća.

Sa druge strane, najveće levo invarijantno fazi kvazi-uređenje na A jednako je:

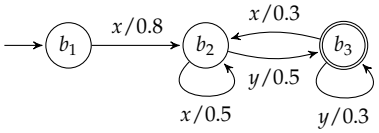
$$\varphi = \varphi^{\mathrm{li}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0.6 & 0 & 1 \end{bmatrix}.$$

Sada, neka je $D = (f, g)$ Mohireva faktorizacija od A, i $D_1 = (f_1, g_1)$ Mohireva faktorizacija od A^D_φ. Imamo da je, na primer, $\varphi \circ \tau_y = [0\ 0.5\ 0.5\ 0.3]$ i $\varphi \circ \tau_{x^k y} =$

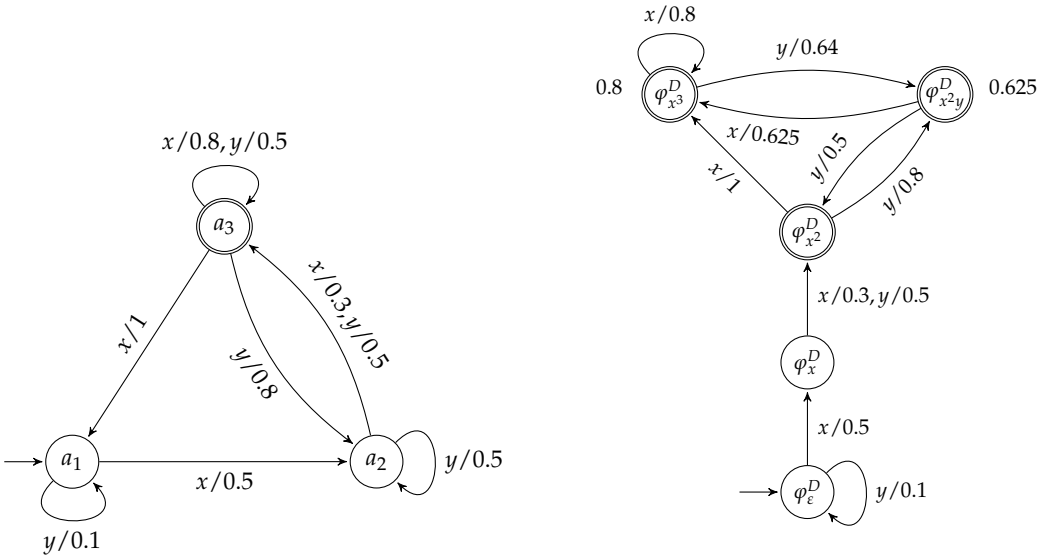


(A) (B)

SLIKA 5.2: KDFA $\tilde{\mathcal{A}}^D$ (a), i $\tilde{\mathcal{A}}^D_\varphi$ (b) konstruisani iz fazi automata $\mathcal{A}$ iz Primera 5.13.



SLIKA 5.3: Graf prelaza fazi automata $\mathcal{B}$ iz Primera 5.13.



SLIKA 5.4: Grafovi prelaza za fazi automate $\mathcal{A}$ (a) i $\mathcal{A}^D_{\varphi^{\mathrm{ri}}}$ (b) iz Primera 5.14

$(1/2)^{k-1} \otimes [0.4\ 0.25\ 0.25\ 0.15] = (1/2)^{k-1} \otimes \varphi \circ \tau_{xy}$ za svako $k \geq 1$. Možemo se uveriti da je za svaki ciklus ispunjen RSSRC. Dakle, KDFA $\tilde{\mathcal{A}}^D_\varphi$ je konačan, i njegov graf prelaza prikazan je na Slici 5.2 (b). Na kraju, graf prelaza KDFA $\mathcal{B} = \widetilde{(\tilde{\mathcal{A}}^D_\varphi)}^{D_1}$ prikazan je na Slici 5.3.

Naredni primer prikazuje FKA koji zadovoljava SSRC, ali ne i RSSRC. Stoga, kanonizaciona metoda koja se sastoji od primene Algoritama 8 i 9 ne može se primeniti jer se ne završava u konačnom broju koraka, dok se Algoritam 6 završava u konačnom broju koraka.

Primer 5.14. Neka je $\mathcal{A}$ FKA nad alfabetom $X = \{x, y\}$ i standardnom proizvod algebrom čiji je graf prelaza prikazan na Slici 5.4 (a).

Imamo da su najveće desno invarijantno i najveće levo invarijantno fazi kvazi-uređenje jednaki, respektivno, sa:

$$\varphi = \varphi^{\mathrm{ri}} = \begin{bmatrix} 1 & 0 & 0 \\ 0.6 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \varphi^{\mathrm{li}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Možemo proveriti da RSSRC nije zadovoljeno, te fazi automat $\mathcal{B}$ dobijen primenom Algoritama 8 i 9 nije konačan. Sa druge strane, SSRC je zadvoljeno, te je graf prelaza KDFA dobijen primenom Algoritma 6 prikazan na Slici 5.4 (b).

Glava 6

Računanje najvećih desno i levo invarijantnih Boolovih matrica i determinizacija težinskih automata

Kao i u slučaju fazi automata, determinizacija nije moguća za sve težinske automate. Drugim rečima, postoje racionalni stepeni redovi koji nisu u stanju da generišu deterministički težinski automat. Pod determinizacijom težinskog automata podrazumevamo algoritam koji za ulazni TKA generiše ekvivalentan KDTA. Uzimajući prethodno rečeno, kao i veliki značaj determinizacije u brojnim oblastima, a posebno u automatskom prepoznavanju govora [32, 154, 155, 167, 168], ne iznenađuje veliki broj radova posvećenih razvoju algoritama za determinizaciju težinskih automata.

Prvi determinizacioni algoritam za težinske automate razvio je Mohri [154] za težinske automate nad tropskim poluprstenom. U istom radu, Mohri dokazuje da je takozvano *svojstvo blizanaca* dovoljan uslov da se njegov algoritam završi u konačnom broju koraka. TKA zadovoljava svojstvo blizanaca ako postoje reči u i v tako da za svaka dva stanja a i b koja su dostižna iz nekog početnog stanja preko reči u, kao i da postoji ciklus pod dejstvom reči v iz oba stanja a i b, tada su težine oba ciklusa jednake. Kao što je primećeno u [6], klasa TKA-a koja zadovoljava svojstvo blizanaca predstavlja široku klasu težinskih automata koji mogu biti determinizovani. Štaviše, provera da li TKA zadovoljava svojstvo blizanaca zahteva polinomijalno vreme [2, 31].

Kirsten i Mäurer [116] adaptirali su Mohriev determinizacioni algoritam za težinske automate nad proizvoljnim poluprstenom, i to čine uvodeći pojmove faktorizacije, koja zavisi od odabranog poluprstena. Takođe, uvode i pojam maksimalne faktorizacije i pokazuju da su najoptimalnije ukoliko je odabrani poluprsten bez delioca nule. Na kraju, pokazuju da je svojstvo blizanaca takođe dovoljan uslov za završetak njihovog determinizacionog algoritma.

U ovoj Glavi dajemo algoritme za determinizaciju težinskih automata nad aditivno idempotentnim i komutativnim poluprstenima bez delioca nule. Ovi determinizacioni metodi predstavljaju adaptaciju determinizacionih metoda za fazi automate prikazanih u prethodnim Glavama. S obzirom da su ti determinizacioni metodi bazirani na određivanju i spajanju ekvivalentnih stanja fazi automata, odnosno na korišćenju desno i levo invarijantnih fazi relacija, potrebno je uvesti odgovarajuće vrste relacija na skupu stanja težinskog automata koje će modelovati ekvivalentnost određenih stanja. Kao što je prikazano u Glavi 1, Boolove matrice mogu poslužiti kao model težinskih relacija ako je poluprsten aditivno idempotentan. Dalje, zahtevaćemo da poluprsten bude bez delioca nule da bi se mogla iskoristiti faktorizacija u determinizacionom procesu. Na kraju, kao što će biti formalno dokazano kasnije,

zahtevaćemo da poluprsten bude komutativan da bismo bili u stanju da poredimo veličinu rezultujućih determinističkih težinskih automata.

Kao i u fazi slučaju, desno i levo invarijantne Boolove matrice u uskoj su vezi sa simulacijama i bisimulacijama na težinskim automatima nad aditivno idempotentnim poluprstenima (videti [51]). Iako su simulacije i bisimulacije za težinske automate definisane i u generalnijim kontekstima (videti [30, 25]), aparatura razvijena u [51] za težinske automate nad aditivno idempotentnim poluprstenima omogućava nam računanje najveće simulacije ili bisimulacije direktnom metodom uz pomoć levog i desnog Boolovog reziduala u konačnom broju koraka. Algoritmi iz [51] za računanje najveće simulacije ili bisimulacije mogu se lako adaptirati za računanje najveće desno i levo invarijantne Boolove matrice. Sa druge strane, u ovoj Glavi pokazujemo da se dodatna poboljšanja mogu dobiti korišćenjem tehnike usitnjenja.

Rezultati iz ove Glave objavljeni su u radu [199], i organizovani su na sledeći način. U Sekciji 6.1 date su definicije desno i levo invarijantnih Boolovih matrica, kao i slabo desno i slabo levo invarijantnih Boolovih matrica. Takođe, prikazani su direktni metodi za računanje najveće desno i levo invarijantne Boolove matrice adaptirani na osnovu algoritama iz [51] za računanje najveće simulacije ili bisimulacije. U Sekciji 6.2 prikazani su algoritmi za računanje najveće desno i levo invarijantne matrica ekvivalencije baziranih na tehnici usitnjenja particija. Dok se direktni metod završava u $\mathcal{O}(mn^5c_+)$ vremenu, pri čemu je $|\mathcal{A}| = n$, $|X| = m$, a c_+ označava vreme izvršenja operacije $+$ u aditivno idempotentnom poluprstenu S, algoritam baziran na tehnici usitnjenja particija izvršava se u $\mathcal{O}(mn^3c_+)$ vremenu. Dalje, u Sekciji 6.3 prikazani su algoritmi za računanje najvećih desno i levo invarijantne matrica kvazi-uređenja, takođe baziranih na tehnici usitnjenja particija. U Sekciji 6.4 dat je kratki prikaz algoritama za računanje najvećih slabo levo i slabo desno invarijantnih matrica ekvivalencije i kvazi-uređenja. Na kraju, u Sekciji 6.5 prikazani su determinizacioni metodi za težinske automate, zajedno sa ilustrativnim primerima.

6.1 Desno i levo invarijantne Boolove matrice

Kroz celu Glavu, S predstavlja aditivno idempotentni poluprsten. Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad S i X. Boolova $A \times A$-matrica $\varrho \in 2^{A\times A}$ nad S naziva se *desno stabilna* ako je:

$$\varrho \cdot \delta_x \cdot \varrho \leq \delta_x \cdot \varrho, \quad \text{za svako } x \in X, \tag{6.1}$$

i naziva se *desno invarijantna* ako je desno stabilna i ako zadovoljava:

$$\varrho \cdot \tau \leq \tau. \tag{6.2}$$

Dualno, Boolova $A \times A$-matrica $\varrho \in 2^{A\times A}$ nad S naziva se *levo stabilna* ako je:

$$\varrho \cdot \delta_x \cdot \varrho \leq \varrho \cdot \delta_x, \quad \text{za svako } x \in X, \tag{6.3}$$

i naziva se *levo invarijantna* ako je levo stabilna i ako zadovoljava:

$$\sigma \cdot \varrho \leq \sigma. \tag{6.4}$$

Za Boolovu $A \times A$-matricu $\varrho \in 2^{A\times A}$ kaže se da je *stabilna* ako je i desno i levo stabilna. Osim toga, Boolova $A \times A$-matrica $\varrho \in 2^{A\times A}$ naziva se *slabo desno invarijantna* ako je:

$$\varrho \cdot \tau_u \leq \tau_u, \quad \text{za svako } u \in X^*. \tag{6.5}$$

dok se naziva *slabo levo invarijantna* ako je:

$$\sigma_u \cdot \varrho \leq \sigma_u, \quad \text{za svako } u \in X^*. \tag{6.6}$$

Indukcijom po dužini reči $u \in X^*$, može se proveriti da je svaka desno (resp. levo) invarijantna matrica i slabo desno (resp. slabo levo) invarijantna. Primetimo da, ukoliko je ϱ refleksivna matrica, tada svi sistemi nejednačina (6.1) – (6.6) postaju sistemi jednačina.

Desno i levo invarijantne matrice u uskoj su vezi sa simulacijama za TKA-e nad aditivno idempotentnim poluprstenima koje su uvedene u [51]. Naime, važi sledeća lema.

Lema 6.1. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka je $\varrho \in 2^{A\times A}$ matrica kvazi-uređenja. Tada su sledeća dva uslova ekvivalentna za svako $x \in X$:*

a) $\varrho \cdot \delta_x \cdot \varrho = \delta_x \cdot \varrho$,

b) $\varrho \cdot \delta_x \leq \delta_x \cdot \varrho$.

Dokaz. Odaberimo proizvoljno $x \in X$. Ako $\varrho \cdot \delta_x \cdot \varrho = \delta_x \cdot \varrho$, tada iz refleksivnosti ϱ dobijamo $\varrho \cdot \delta_x \leq \varrho \cdot \delta_x \cdot \varrho = \delta_x \cdot \varrho$. Obratno, ako je $\varrho \cdot \delta_x \leq \delta_x \cdot \varrho$, s obzirom da je ϱ matrica kvazi-uređenja imamo da je $\varrho \cdot \delta_x \cdot \varrho \leq \delta_x \cdot \varrho \cdot \varrho = \delta_x \cdot \varrho$. Druga nejednakost sledi iz refleksivnosti ϱ. □

Iz Leme 6.1 zaključujemo da je matrica kvazi-uređenja $\varrho \in 2^{A\times A}$ desno invarijantna ako je njena transponovana matrica ϱ^{T} direktna simulacija na $\mathcal{A}$. Slično se pokazuje da je matrica kvazi-uređenja levo invarijantna ako je povratna simulacija na $\mathcal{A}$.

Teorema 6.2. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA. Tada postoje najveće desno i levo invarijantne (resp. slabo desno i slabo levo invarijantne) matrice kvazi-uređenja.*

Dokaz. S obzirom da je identička matrica I_A desno invarijantna matrica kvazi-uređenja, familija $\{\varrho_i\}_{i\in I}$ svih desno invarijantnih matrica kvazi-uređenja je neprazna. Koristeći Lemu 6.1, zaključujemo da je suma elemenata familije $\{\varrho_i\}_{i\in I}$ takođe desno invarijantna matrica, ali ne obavezno i matrica kvazi-uređenja. Sa druge strane, s obzirom da je drugi stepen (a samim tim i prozivoljni stepen) desno invarijantne matrice kvazi-uređenja takođe desno invarijantna matrica kvazi-uređenja, zaključujemo da je tranzitivno zatvorenje sume elemenata familije $\{\varrho_i\}_{i\in I}$, definisano sa:

$$\varrho^{\mathrm{ri}} = \sum_{n\in\mathbb{N}} \big(\sum_{i\in I} \varrho_i\big)^n,$$

desno invarijantna matrica kvazi-uređenja. Dakle, ϱ^{ri} je najveća desno invarijantna matrica kvazi-uređenja. Slično se tvrđenje pokazuje i za najveću levo invarijatnu matricu kvazi-uređenja, a isti argumenti važe i za slabo invarijantne matrice. □

Teorema 6.2 obezbeđuje postojanje najveće desno (ili levo) invarijantne matrice kvazi-uređenja (kao i matrice ekvivalencije), ali ne daje način za njeno računanje. Međutim, odgovarajući algoritmi mogu se lako izvesti iz algoritama za računanje najvećih simulacija između dva TKA datih u [51]. U narednoj Teoremi navodimo načine za računanje najvećih desno i levo invarijantnih matrica kvazi-uređenja i ekvivalencija, a koji se zasnivaju na računanju desnih i levih reziduala za Boolove matrice (videti (1.68) i (1.69)). S obzirom da naredna Teorema predstavlja varijaciju [51, Teorema 5.4], njen dokaz je izostavljen.

Teorema 6.3. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka je $\{\varrho_k\}_{k\in\mathbb{N}} \subseteq 2^{A\times A}$ niz Boolovih $A \times A$-matrica definisan induktivno sa:*

$$\varrho_1 = \tau/\tau \quad i \quad \varrho_{k+1} = \varrho_k \odot \bigodot_{x\in X} (\delta_x \cdot \varrho_k)/(\delta_x \cdot \varrho_k), \quad \text{za svako } k \in \mathbb{N}.$$

Tada je niz $\{\varrho_k\}_{k\in\mathbb{N}}$ konačan i opadajući, i stabilizuje se za neko $m \in \mathbb{N}$, tj. postoji $m \in \mathbb{N}$ tako da je $\varrho_m = \varrho_{m+1}$. Štaviše, ϱ_m je najveća desno invarijantna matrica kvazi-uređenja.

Na sličan način računamo najveću levo invarijantnu matricu kvazi-uređenja, tj. jedina razlika je da se niz $\{\varrho_k\}_{k\in\mathbb{N}} \subseteq 2^{A\times A}$ računa uz pomoć Boolovog desnog reziduala (1.69) na sledeći način:

$$\varrho_1 = \tau\backslash\tau \quad \text{i} \quad \varrho_{k+1} = \varrho_k \odot \bigodot_{x\in X} (\varrho_k \cdot \delta_x)\backslash(\varrho_k \cdot \delta_x), \quad \text{za svako } k \in \mathbb{N},$$

kao i desno i levo invarijantne matrice ekvivalencije, pri čemu se niz $\{\varrho_k\}_{k\in\mathbb{N}} \subseteq 2^{A\times A}$ idnuktivno konstruiše na jedan od odgovarajućih načina:

$$\varrho_1 = \tau|\tau \quad \text{i} \quad \varrho_{k+1} = \varrho_k \odot \bigodot_{x\in X} (\delta_x \cdot \varrho_k)|(\delta_x \cdot \varrho_k), \quad \text{za svako } k \in \mathbb{N}, \text{ odnosno}$$
$$\varrho_1 = \tau|\tau \quad \text{i} \quad \varrho_{k+1} = \varrho_k \odot \bigodot_{x\in X} (\varrho_k \cdot \delta_x)|(\varrho_k \cdot \delta_x), \quad \text{za svako } k \in \mathbb{N}.$$

Odgovarajući algoritmi za računanje najveće desno ili levo invarijantne matrice kvazi-uređenja ili ekvivalencije mogu se lako izvesti na osnovu prethodnih formula. Svi algoritmi završavaju se u konačnom broju koraka i rade u polinomijalnoj vremenskoj složenosti, odnosno u $\mathcal{O}(mn^5c_+)$ vremenu, gde je $|\mathcal{A}| = n$, $|X| = m$, i c_+ označava vreme izvršenja operacije $+$ u aditivno idempotentnom poluprstenu S. (uporediti sa [51]).

6.2 Računanje najvećih desno i levo invarijantnih matrica ekvivalencije

Kao što smo već napomenuli, algoritmi za računanje najvećih desno i levo invarijantnih matrica ekvivalencije mogu se poboljšati tehnikom usitnjenja particija. Preciznije, koristimo tehniku Paige i Tarjana [160], te za razliku od direktnog metoda zadatog preko Teoreme 6.3, pamtimo i dodatnu particiju uz trenutnu particiju, tako da su određena svojstva zadovoljena. U onome što sledi, dajemo formalni opis algoritma.

Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka su $\varrho, \theta \in 2^{A\times A}$ matrice ekvivalencije. Tada kažemo da je ϱ *stabilna u odnosu na* θ ako je:

$$\varrho \leq \bigodot_{x\in X} (\delta_x \cdot \theta^a)|(\delta_x \cdot \theta^a), \quad \text{za svako } a \in A. \tag{6.7}$$

U slučaju kada je $\mathcal{A}$ TKA nad Boolovim poluprstenom, prethodna definicija poklapa se sa definicijom stabilnosti particija na skupu stanja NKA (uporediti sa [1, 48, 160]).

Naredna Lema daje vezu između stabilnosti matrica ekvivalencija na fazi automatu i njihovih stabilnosti u odnosu na druge matrice.

Lema 6.4. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA i $\varrho \in 2^{A\times A}$ matrica ekvivalencije. Tada je ϱ stabilna u odnosu na samu sebe ako je desno stabilna na $\mathcal{A}$.*

Dokaz. Prema (1.73), $\varrho \in 2^{A\times A}$ je desno stabilna akko

$$\varrho \cdot \delta_x \cdot \varrho^a \leq \delta_x \cdot \varrho^a, \quad \text{za svako } a \in A \text{ i } x \in X,$$

ili ekvivalentno,

$$\varrho \leq (\delta_x \cdot \varrho^a)/(\delta_x \cdot \varrho^a), \quad \text{za svako } a \in A \text{ i } x \in X.$$

Dalje, s obzirom da je ϱ simetrična matrica, imamo i da je:

$$\varrho = \varrho^{\mathrm{T}} \leq ((\delta_x \cdot \varrho^a)/(\delta_x \cdot \varrho^a))^{\mathrm{T}} = (\delta_x \cdot \varrho^a)\backslash(\delta_x \cdot \varrho^a), \quad \text{za svako } a \in A \text{ i } x \in X.$$

čime je dokaz završen. □

Lema 6.5. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA i neka su $\varrho, \theta \in 2^{A\times A}$ matrice ekvivalencije. Ako je $\varrho \leq \theta$ i ϱ desno stabilna, tada je ϱ stabilna u odnosu na θ.*

Dokaz. S obzirom da ϱ zadovoljava (6.1), imamo da je:

$$\varrho \cdot \delta_x \cdot \varrho \cdot \theta \leq \delta_x \cdot \varrho \cdot \theta, \quad \text{za svako } x \in X,$$

i prema Lemi 1.33 dalje imamo da je:

$$\varrho \cdot \delta_x \cdot \theta \leq \delta_x \cdot \theta, \quad \text{za svako } x \in X.$$

Koristeći (1.73), zaključujemo da važi:

$$\varrho \cdot \delta_x \cdot \theta^a \leq \delta_x \cdot \theta^a, \quad \text{za svako } x \in X \text{ i } a \in A,$$

što je ekvivalentno sa:

$$\varrho \leq (\delta_x \cdot \theta^a)/(\delta_x \cdot \theta^a), \quad \text{za svako } x \in X \text{ i } a \in A.$$

Štaviše, iz simetričnosti ϱ dobijamo:

$$\varrho = \varrho^{\mathrm{T}} \leq ((\delta_x \cdot \theta^a)/(\delta_x \cdot \theta^a))^{\mathrm{T}} = (\delta_x \cdot \theta^a)\backslash(\delta_x \cdot \theta^a), \quad \text{za svako } x \in X \text{ i } a \in A.$$

Dakle, $\varrho \leq (\delta_x \cdot \theta^a)|(\delta_x \cdot \theta^a)$, za svako $x \in X$ i $a \in A$, odakle sledi (6.7). □

Sada se navodi metod za računanje najveće desno invarijantne matrice ekvivalencije za dati TKA.

Teorema 6.6. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka su $\{\theta_k\}_{k\in\mathbb{N}}$ i $\{\varrho_k\}_{k\in\mathbb{N}}$ dva niza Boolovih $A \times A$-matrica induktivno definisanih na sledeći način: Za $k = 1$, postavimo da je:*

$$\theta_1 = U_A, \tag{6.8}$$

$$\varrho_1 = \tau|\tau \odot \bigodot_{x\in X}(\delta_x \cdot \theta_1^a)|(\delta_x \cdot \theta_1^a), \quad \textit{za neko } a \in A. \tag{6.9}$$

Pretpostavimo da smo izračunali θ_k i ϱ_k u trenutnom koraku k. U sledećem koraku, ako je $\theta_k \neq \varrho_k$, odaberimo neko $a \in A$ za koje je $\theta_k^a \neq \varrho_k^a$, i izračunajmo θ_{k+1} i ϱ_{k+1} na sledeći

način:

$$\theta_{k+1} = \theta_k \odot (\varrho_k^a | \varrho_k^a), \tag{6.10}$$

$$\varrho_{k+1} = \varrho_k \odot \bigodot_{x \in X} \big((\delta_x \cdot \varrho_k^a) | (\delta_x \cdot \varrho_k^a) \odot (\delta_x \cdot (\theta_k^a - \varrho_k^a)) | (\delta_x \cdot (\theta_k^a - \varrho_k^a))\big). \tag{6.11}$$

U suprotnom, kada je $\theta_k = \varrho_k$, postavimo da je $\theta_k = \theta_{k+1}$ i $\varrho_k = \varrho_{k+1}$. Tada važe sledeća tvrđenja:

a) Nizovi $\{\theta_k\}_{k\in\mathbb{N}}$ i $\{\varrho_k\}_{k\in\mathbb{N}}$ su opadajući;

b) ϱ_k i θ_k su matrice ekvivalencije, za svako $k \in \mathbb{N}$;

c) ϱ_k je usitnjenje od θ_k (tj. $\varrho_k \leq \theta_k$), za svako $k \in \mathbb{N}$;

d) ϱ_k je stabilna u odnosu na θ_k, za svako $k \in \mathbb{N}$;

e) Postoji $s \in \mathbb{N}$ tako da je $\theta_s = \varrho_s$, i ϱ_s je najveća desno invarijantna matrica ekvivalencije na A.

Dokaz. Delovi a) i b) slede direktno iz definicija θ_k i ϱ_k;

c) Dokazujemo da je $\varrho_k \leq \theta_k$ za svako $k \in \mathbb{N}$ indukcijom po k. U slučaju $k = 1$ imamo da je $\varrho_1 \leq U_A = \theta_1$. Pretpostavimo da je $\varrho_m \leq \theta_m$ za neko $k = m$. Prema a) imamo da je $\varrho_{m+1} \leq \varrho_m$. Na osnovu Leme 1.35 imamo da je $\varrho_m \leq \varrho_m^c | \varrho_m^c$ za svako $c \in A$, i kombinujući sa indukcijskom hipotezom $\varrho_m \leq \theta_m$, dobijamo da je $\varrho_m \leq \theta_m \odot (\varrho_m^a | \varrho_m^a) = \theta_{m+1}$. Stoga je $\varrho_{m+1} \leq \varrho_m \leq \theta_{m+1}$, što je i trebalo pokazati.

d) Dokazujemo da:

$$\varrho_k \leq \bigodot_{x \in X} (\delta_x \cdot \theta_k^a) | (\delta_x \cdot \theta_k^a), \quad \text{za svako } a \in A, \tag{6.12}$$

važi za svako $k \in \mathbb{N}$ indukcijom po k. Direktno iz (6.9) možemo zaključiti da je (6.12) validno za $k = 1$.

U slučaju kada je $k = 2$ imamo da je $\theta_2 = \theta_1 \odot (\varrho_1^a | \varrho_1^a)$ za neko $a \in A$. S obzirom da θ_1 ima tačno jednu klasu ekvivalencije $\theta_1^a = 1^A$, to znači da se θ_2 dobija iz θ_1 deljenjem klase ekvivalencije θ_1^a na dve klase ekvivalencije ϱ_1^a i $\theta_1^a - \varrho_1^a$. Prema (6.11) imamo da je:

$$\varrho_2 = \varrho_1 \odot \bigodot_{x \in X} \big((\delta_x \cdot \varrho_1^a) | (\delta_x \cdot \varrho_1^a) \odot (\delta_x \cdot (\theta_1^a - \varrho_1^a)) | (\delta_x \cdot (\theta_1^a - \varrho_1^a))\big),$$

što znači da je:

$$\varrho_2 \leq \bigodot_{x \in X} (\delta_x \cdot \varrho_1^a) | (\delta_x \cdot \varrho_1^a)$$

kao i:

$$\varrho_2 \leq \bigodot_{x \in X} (\delta_x \cdot (\theta_1^a - \varrho_1^a)) | (\delta_x \cdot (\theta_1^a - \varrho_1^a)),$$

odnosno, (6.12) važi za obe klase ekvivalencije od θ_2, što znači da (6.12) važi u slučaju kada je $k = 2$.

Pretpostavimo sada da je (6.12) validno za neko $k = m$. Posmatrajmo matricu ekvivalencije $\theta_{m+1} = \theta_m \odot (\varrho_m^a | \varrho_m^a)$ za neko $a \in A$ tako da je $\theta_m^a \neq \varrho_m^a$. Iz c) sledi da je $\varrho_m^a < \theta_m^a$, što znači da se θ_{m+1} dobija iz θ_m deljenjem klase ekivivalencije θ_m^a na dve klase ekvivalencije ϱ_m^a i $\theta_m^a - \varrho_m^a$ (dok sve ostale klase ekvivalencije ostaju iste). Stoga, za svaku klasu ekvivalencije $\theta_{m+1}^b \in \theta_{m+1}$, $b \in A$, takvu da je $\theta_{m+1}^b \neq \varrho_m^a$ i

$\theta^b_{m+1} \neq \theta^a_m - \varrho^a_m$, prema a) i indukcijskoj pretpostavci imamo da je:

$$\varrho_{m+1} \leq \varrho_m \leq \bigodot_{x\in X}(\delta_x \cdot \theta^b_m)|(\delta_x \cdot \theta^b_m) = \bigodot_{x\in X}(\delta_x \cdot \theta^b_{m+1})|(\delta_x \cdot \theta^b_{m+1}).$$

Drugim rečima, (6.12) važi za svako $b \in A$ za koje je $\theta^b_{m+1} \neq \varrho^a_m$ i $\theta^b_{m+1} \neq \theta^a_m - \varrho^a_m$. Sa druge strane, direktno iz (6.11) imamo da je:

$$\varrho_{m+1} = \varrho_m \odot \bigodot_{x\in X}\big((\delta_x \cdot \varrho^a_m)|(\delta_x \cdot \varrho^a_m) \odot (\delta_x \cdot (\theta^a_m - \varrho^a_m))|(\delta_x \cdot (\theta^a_m - \varrho^a_m))\big)$$

što znači da je:

$$\varrho_{m+1} \leq \bigodot_{x\in X}(\delta_x \cdot \varrho^a_m)|(\delta_x \cdot \varrho^a_m)$$

kao i:

$$\varrho_{m+1} \leq \bigodot_{x\in X}(\delta_x \cdot (\theta^a_m - \varrho^a_m))|(\delta_x \cdot (\theta^a_m - \varrho^a_m)),$$

odnosno, (6.12) važi u slučaju $k = m + 1$, što je i trebalo pokazati.

e) S obzirom da je A konačan skup, imamo da je $2^{A\times A}$ takođe konačan skup, stoga se oba niza $\{\theta_k\}_{k\in\mathbb{N}}$ i $\{\varrho_k\}_{k\in\mathbb{N}}$ stabilizuju. Ali, ukoliko za neko $s \in \mathbb{N}$ imamo da je $\theta_s = \theta_{s+1}$, prema konstrukciji ova dva niza imamo da važi i $\theta_s = \varrho_s$.
Ostaje da pokažemo da je ϱ_s najveća desno invarijantna matrica ekvivalencije. S obzirom da iz d) sledi da je ϱ_s stabilna u odnosu na θ_s, kao i $\theta_s = \varrho_s$, dobijamo da je ϱ_s stabilna, tj.

$$\varrho_s \leq \bigodot_{x\in X}(\delta_x \cdot \varrho^c_s)|(\delta_x \cdot \varrho^c_s),$$

za svako $c \in A$, što znači da ϱ_s zadovoljava (6.1). Takođe, iz a) imamo da je $\varrho_s \leq \varrho_1 \leq \tau|\tau$, stoga, ϱ_s takođe zadovoljava i (6.2) te je ϱ_s desno invarijantna matrica ekvivalencije. Da bi dokazali da je ϱ_s ujedno i najveća, dokazujemo da je $\omega \leq \varrho_k$ za svaku desno invarijantnu matricu ekvivalencije $\omega \in 2^{A\times A}$ indukcijom po k.
Za $k = 1$, imamo da je $\omega \leq U_A = \theta_1$, i s obzirom da je ω desno stabilna, iz Leme 6.5 imamo da je ω stabilna u odnosu na θ. Takođe, ω zadovolajva (6.2), što znači da je:

$$\omega \leq \tau/\tau \quad \text{i} \quad \omega = \omega^{\mathrm{T}} \leq (\tau/\tau)^{\mathrm{T}} = \tau\backslash\tau,$$

dakle, $\omega \leq \tau|\tau$, stoga je:

$$\omega \leq \tau|\tau \odot \bigodot_{x\in X}(\delta_x \cdot \theta^a_1)|(\delta_x \cdot \theta^a_1) = \varrho_1.$$

Pretpostavimo sada da je $\omega \leq \varrho_m$ za neko $k = m$. Iz b) imamo da je $\omega \leq \theta_m$, a iz Leme 1.35 takođe imamo da je $\omega \leq \varrho^a_m|\varrho^a_m$, pri čemu je $a \in A$ takvo da je $\varrho^a_m \neq \theta^a_m$. Stoga,

$$\omega \leq \theta_m \odot (\varrho^a_m|\varrho^a_m) = \theta_{m+1}.$$

S obzirom da je ω desno stabilna, iz Leme 6.5 sledi da je ω stabilna u odnosu na θ_{m+1}, tj. (6.12) važi za sve klase ekvivalencije od θ_{m+1}, i posebno za klase ekvivalencije ϱ^a_m i $\theta^a_m - \varrho^a_m$. To znači da je:

$$\omega \leq \varrho_m \odot \bigodot_{x\in X}\big((\delta_x \cdot \varrho^a_m)|(\delta_x \cdot \varrho^a_m) \odot (\delta_x \cdot (\theta^a_m - \varrho^a_m))|(\delta_x \cdot (\theta^a_m - \varrho^a_m))\big) = \varrho_{m+1},$$

što je i trebalo dokazati. □

Prethodna Teorema lako se može transformisati u algoritam za računanje najveće desno invarijantne matrice ekvivalencije za dati TKA nad aditivno idempotentnim poluprstenom.

Algoritam 10 Računanje najveće desno invarijantne matrice ekvivalencije na težinskom automatu

Ulaz: TKA $\mathcal{A} = (A, \delta, \sigma, \tau)$ nad alfabetom X i aditivno idempotentnom poluprstenu S

Izlaz: Najveća desno invarijantna matrica ekvivalencije

1: $\theta \leftarrow U_A$
2: $\varrho \leftarrow \tau | \tau \odot \bigodot_{x \in X} (\delta_x \cdot \theta^a) | (\delta_x \cdot \theta^a), \quad$ za neko $a \in A$
3: **while** $\theta \neq \varrho$ **do**
4: Izaberimo $a \in A$ tako da je $\theta^a \neq \varrho^a$
5: $\theta_1 \leftarrow \theta \odot (\varrho^a | \varrho^a)$
6: $\varrho_1 \leftarrow \varrho \odot \bigodot_{x \in X} \big((\delta_x \cdot \varrho^a) | (\delta_x \cdot \varrho^a) \odot (\delta_x \cdot (\theta^a - \varrho^a)) | (\delta_x \cdot (\theta^a - \varrho^a)) \big)$
7: $\theta \leftarrow \theta_1, \quad \varrho \leftarrow \varrho_1$
8: **end while**
9: **return** ϱ

Prema Teoremi 6.6, nizovi $\{\theta_k\}_{k \in \mathbb{N}}$ i $\{\varrho_k\}_{k \in \mathbb{N}}$ su opadajući i stabilizuju se, stoga se Algoritam 10 uvek završava u konačnom broju koraka, bez obzira na odabir aditivno idempotentnog poluprstena.

Odredimo sada vreme izvršenja Algoritma 10. Označimo, kao i ranije, $|\mathcal{A}| = n$ i $|X| = m$. Takđe, neka c_+ i $c_\cdot$ označavaju, redom, vreme izračunavanja operacija $+$ i $\cdot$ u S. Na primer, kada je S Boolov, tropski ili Viterbiev poluprsten, tada je $c_+ = c_\cdot = 1$.

Korak 1 izvršava se u $\mathcal{O}(n^2)$ vremenu, dok Korak 2 zahteva izračunavanje Hadamardovog prozivoda $m + 1$ Boolovih matrica, pri čemu je svaka Boolova matrica izračunata preko formule (1.72). S obzirom da elementi Boolove matrice mogu biti jedino nula i jedinica iz poluprstena, množenje se završava u konstantnom vremenu, pa je stoga za izračunavanje svakog A-vektora $\delta_x \cdot \theta^a$ potrebno $\mathcal{O}(n^2 c_+)$ vreme. Izračunavanje matrice zadate preko formule (1.72) zahteva n^2 provera jednakosti, te stoga zahteva $\mathcal{O}(n^2)$ vreme. Dakle, Korak 2 se završava u $\mathcal{O}(mn^2 c_+)$ vremenu.

U Koraku 4 imamo, u najgorem slučaju, prolazak kroz sve klase ekvivalencije matrica θ i ϱ, a svaka može imati najviše n klasa ekvivalencije. Provera da li su dve klase ekvivalencije jednake zahteva $\mathcal{O}(n)$ vreme, dakle Korak 4 izvršava se u $\mathcal{O}(n^2)$ vremenu. Slično kao u analizi Koraka 2, zaključujemo da se Korak 5 izvršava u $\mathcal{O}(n^2)$ vremenu, a Korak 6 u $\mathcal{O}(mn^2 c_+)$ vremenu. Očigledno je da se Korak 7 izvršava u $\mathcal{O}(n^2)$ vremenu.

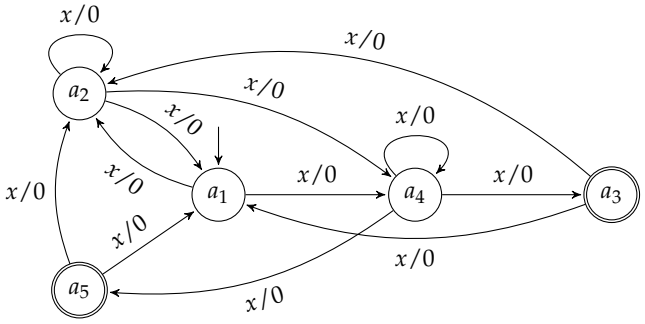
Prema Teoremi 6.6, ukupan broj ponavljanja Koraka 4-7 je najviše $n - 1$. Stoga je ukupno vreme izvršenja Algoritma 10 jednako $\mathcal{O}(mn^3 c_+)$. Dakle, Algoritam se izvršava u polinomijalnom vremenu koje je brže od vremena koje zahteva direktan metod zadat preko Teoreme 6.3.

Algoritam za računanje najveće levo invarijantne matrice ekvivalencije može se analogno izvesti, te ga izostavljamo.

Primer 6.7. Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad tropskim poluprstenom pri čemu je $A = \{a_1, a_2, a_3, a_4, a_5\}$ i $X = \{x\}$ čiji je graf prelaza dat na Slici 6.1. Imamo da su početni težinski vektor, završni težinski vektor, kao i težinske matrice prelaza zadate

sa:

$$\sigma = \begin{bmatrix} 0 & \infty & \infty & \infty & \infty \end{bmatrix}, \quad \tau = \begin{bmatrix} \infty \\ \infty \\ 0 \\ \infty \\ 0 \end{bmatrix}, \quad d_x = \begin{bmatrix} \infty & 0 & \infty & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 & 0 \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}.$$



SLIKA 6.1: The transition graph of the WFA from Example 6.7

U prvom koraku, Algoritam 10 daje matrice:

$$\theta_1 = U_A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \varrho_1 = \tau | \tau \odot (\delta_x \cdot \theta_1^a) | (\delta_x \cdot \theta_1^a) = \begin{bmatrix} 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix}.$$

U sledećem koraku, za $a_1 \in A$ imamo da je $\varrho_1^{a_1} \neq \theta_1^{a_1}$, stoga odaberimo $a_1 \in A$ i dobijamo:

$$\theta_2 = \theta_1 \odot (\varrho_1^{a_1} | \varrho_1^{a_1}) = \begin{bmatrix} 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix},$$

$$\varrho_2 = \varrho_1 \odot (\delta_x \cdot \varrho_1^{a_1}) | (\delta_x \cdot \varrho_1^{a_1}) \odot (\delta_x \cdot (\theta_1^{a_1} - \varrho_1^{a_1})) | (\delta_x \cdot (\theta_1^{a_1} - \varrho_1^{a_1})) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix}.$$

Ponovo, odaberimo $a_1 \in A$ jer je $\varrho_2^{a_1} \neq \theta_2^{a_1}$, i dobijamo:

$$\theta_3 = \theta_2 \odot (\varrho_2^{a_1} | \varrho_2^{a_1}) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix},$$

$$\varrho_3 = \varrho_2 \odot (\delta_x \cdot \varrho_2^{a_1})|(\delta_x \cdot \varrho_2^{a_1}) \odot (\delta_x \cdot (\theta_2^{a_1} - \varrho_2^{a_1}))|(\delta_x \cdot (\theta_2^{a_1} - \varrho_2^{a_1})) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix}.$$

S obzirom da je $\theta_3 = \varrho_3$, zaključujemo da je najveća desno invarijantna matrica ekvivalencije jednaka $\theta_3 = \varrho_3$.

6.3 Računanje najvećih desno i levo invarijantnih matrica kvazi-uređenja

Algoritmi dati u prethodnoj Sekciji mogu se generalizovati za računanje najvećih desno i levo invarijantnih matrica kvazi-uređenja.

Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka su $\varrho, \theta \in 2^{A \times A}$ matrice kvazi-uređenja. Tada kažemo da je ϱ *desno stabilna u odnosu na* θ ako je:

$$\varrho \leq \bigodot_{x \in X} (\delta_x \cdot \theta a)/(\delta_x \cdot \theta a), \quad \text{za svako } a \in A,$$

i dualno, ϱ je *levo stabilna u odnosu na* θ ako je:

$$\varrho \leq \bigodot_{x \in X} (a\theta \cdot \delta_x)\backslash(a\theta \cdot \delta_x), \quad \text{za svako } a \in A.$$

Naredne dve Leme predstavljaju direktnu generalizaciju Lema 6.4 i 6.5.

Lema 6.8. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka su $\varrho \in 2^{A \times A}$ matrice kvazi-uređenja. Tada je ϱ desno stabilna (odnosno, levo stabilna) u odnosu na samu sebe akko je desno stabilna (odnosno, levo stabilna) na $\mathcal{A}$.*

Lema 6.9. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA i $\varrho, \theta \in 2^{A \times A}$ matrice kvazi-uređenja. Ako je $\varrho \leq \theta$ i ϱ desno stabilna (odnosno, levo stabilna), tada je ϱ desno stabilna (odnosno, levo stabilna) u odnosu na θ.*

Naredna Teorema formalizuje način za računanje najveće desno invarijantne matrice kvazi-uređenja.

Teorema 6.10. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA, i neka su $\{\theta_k\}_{k \in \mathbb{N}}$ i $\{\varrho_k\}_{k \in \mathbb{N}}$ dva niza Boolovih $A \times A$-matrica induktivno definisanih na sledeći način: Za $k = 1$, stavimo:*

$$\theta_1 = U_A,$$
$$\varrho_1 = \tau/\tau \odot \bigodot_{x \in X} (\delta_x \cdot \theta_1 a)/(\delta_x \cdot \theta_1 a), \quad \text{za neko } a \in A.$$

Pretpostavimo da smo izračunali θ_k i ϱ_k u trenutnom koraku k. U sledećem koraku, ako je $\varrho_k \neq \theta_k$, odaberimo neko $a \in A$ tako da je $\theta_k a \neq \varrho_k a$, i izračunajmo skup $B = \{b \in A | \varrho_k a(b) \neq 0\}$. Potom izračunajmo θ_{k+1} i ϱ_{k+1} na sledeći način:

$$\theta_{k+1} = \theta_k \odot (\varrho_k a / \varrho_k a),$$
$$\varrho_{k+1} = \varrho_k \odot \bigodot_{x \in X} \bigodot_{b \in B} (\delta_x \cdot \theta_{k+1} b)/(\delta_x \cdot \theta_{k+1} b).$$

Inače, postavimo da je $\theta_{k+1} = \theta_k$ i $\varrho_{k+1} = \varrho_k$. Tada važi:

a) Nizovi $\{\theta_k\}_{k\in\mathbb{N}}$ *i* $\{\varrho_k\}_{k\in\mathbb{N}}$ *su opadajući;*

b) ϱ_k *i* θ_k *su matrice ekvivalencije, za svako* $k \in \mathbb{N}$*;*

c) ϱ_k *je usitnjenje od* θ_k*, za svako* $k \in \mathbb{N}$*;*

d) ϱ_k *je desno stabilna u odnosu na* θ_k*, za svako* $k \in \mathbb{N}$*;*

e) Postoji $s \in \mathbb{N}$ *tako da je* $\theta_s = \varrho_s$*, i* ϱ_s *je najveća desno invarijantna matrica kvazi-uređenja.*

Dokaz. Delovi a), b), c) i e) mogu se dokazati analogno kao u Teoremi 6.6 koristeći Leme 6.8 i 6.9.

d) Dokazujemo da

$$\varrho_k \leq \bigodot_{x\in X} (\delta_x \cdot \theta_k a)/(\delta_x \cdot \theta_k a), \quad \text{za svako } a \in A, \tag{6.13}$$

važi za svako $k \in \mathbb{N}$ indukcijom po k. U slučaju $k = 1$, (6.13) sledi direktno iz definicije ϱ_1 jer se skup A/θ_1 sastoji od jednog elementa (foreseta) $\theta_1^a = 1^A$. Pretpostavimo da (6.13) važi za neko $k = m$ te dokažimo da takođe važi i za $m + 1$. Prema definiciji ϱ_{m+1} imamo da je

$$\varrho_k \leq \bigodot_{x\in X} (\delta_x \cdot \theta_k c)/(\delta_x \cdot \theta_k c), \quad \text{za svako } c \in B. \tag{6.14}$$

Dokažimo sada da je $\theta_m c = \theta_{m+1} c$ za svako $c \in A - B$. Ako je $c \in A - B$, tada važi $\varrho_m a(c) = 0$ te stoga, za svako $d \in A$, važi $\varrho_m a(c) \leq \varrho_m a(d)$, kao i:

$$\begin{aligned}\theta_{m+1}c(d) &= \theta_{m+1}(d,c) = \theta_m(d,c) \cdot (\varrho_m a/\varrho_m a)(d,c)\\ &= \theta_m(d,c) \cdot 1 = \theta_m(d,c) = \theta_m c(d).\end{aligned}$$

Prethodno važi za svako $d \in A$, te prema indukcijskoj pretpostavci imamo da je:

$$\varrho_{m+1} \leq \varrho_m \leq \bigodot_{x\in X} (\delta_x \cdot \theta_m c)/(\delta_x \cdot \theta_m c) = \bigodot_{x\in X} (\delta_x \cdot \theta_{m+1} c)/(\delta_x \cdot \theta_{m+1} c), \tag{6.15}$$

za svako $c \in A - B$. Kombinujući (6.14) i (6.15), (6.13) sledi u slučaju $k = m + 1$, čime je dokaz završen. □

Prethodna Teorema lako se transformiše u Algoritam za računanje najveće desno invarijantne matrice kvazi-uređenja za dati TKA nad aditivno idempotentnim poluprstenom.

Kao i kod Algoritma 10, Algoritam 11 uvek se završava u konačnom broju koraka, nezavisno od odabira aditivno idempotentnog poluprstena.

Primer 6.11. Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA iz Primera 6.7. Cilj nam je izračunavanje najveće desno invarijantne matrice kvazi-uređenja preko Algoritma 11. U prvom koraku dobijamo matrice:

$$\theta_1 = U_A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; \varrho_1 = \tau/\tau \odot (\delta_x \cdot \theta_1 a)/(\delta_x \cdot \theta_1 a) = \begin{bmatrix} 0 & 0 & \infty & 0 & \infty\\ 0 & 0 & \infty & 0 & \infty\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & \infty & 0 & \infty\\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Algoritam 11 Računanje najveće desno invarijantne matrice kvazi-uređenja na težinskom automatu

Ulaz: TKA $\mathcal{A} = (A, \delta, \sigma, \tau)$ nad alfabetom X i aditivno idempotentnim poluprstenom S

Izlaz: Najveća desno invarijantna matrica kvazi-uređenja

1: $\theta \leftarrow U_A$
2: $\varrho \leftarrow \tau / \tau \odot \bigodot_{x \in X} (\delta_x \cdot \theta a) / (\delta_x \cdot \theta a)$, za neko $a \in A$
3: **while** $\theta \neq \varrho$ **do**
4: Izaberimo $a \in A$ tako da je $\theta a \neq \varrho a$
5: Izračunati $B = \{b \in A | \varrho a(b) \neq 0\}$
6: $\theta_1 \leftarrow \theta \odot (\varrho a / \varrho a)$
7: $\varrho_1 \leftarrow \varrho \odot \bigodot_{x \in X} \bigodot_{b \in B} (\delta_x \cdot \theta_1 b) / (\delta_x \cdot \theta_1 b)$
8: $\theta \leftarrow \theta_1$, $\varrho \leftarrow \varrho_1$
9: **end while**
10: **return** ϱ

U sledećem koraku odaberimo, na primer, $a_3 \in A$, jer je $\theta_1 a_3 \neq \varrho_1 a_3$, i izračunajmo skup $B = \{b \in A | \varrho_1 a_3(b) \neq 0\} = \{a_3, a_5\}$. Tada imamo:

$$\theta_2 = \theta_1 \odot (\varrho_1 a_3 / \varrho_1 a_3) = \begin{bmatrix} 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

S obzirom da je $\theta_2 a_3 = \theta_2 a_5$, dalje imamo:

$$\varrho_2 = \varrho_1 \odot (\delta_x \cdot \theta_2 a_3) / (\delta_x \cdot \theta_2 a_3) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & 0 & \infty & 0 \end{bmatrix}.$$

U sledećem korako, s obzirom da za $a_4 \in A$ imamo da je $\theta_1 a_4 \neq \varrho_1 a_4$, odaberimo $a_4 \in A$, i izračunajmo skup $B = \{b \in A | \varrho_2 a_4(b) \neq 0\} = \{a_4\}$. Tada imamo:

$$\theta_3 = \theta_2 \odot (\varrho_2 a_4 / \varrho_2 a_4) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ 0 & 0 & 0 & \infty & 0 \end{bmatrix},$$

$$\varrho_3 = \varrho_2 \odot (\delta_x \cdot \theta_3 a_4) / (\delta_x \cdot \theta_3 a_4) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix}.$$

Na kraju, odaberimo $a_1 \in A$, i dobijamo:

$$\theta_4 = \theta_3 \odot (\varrho_3 a_1 / \varrho_3 a_1) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix},$$

$$\varrho_4 = \varrho_3 \odot (\delta_x \cdot \theta_4 a_1) / (\delta_x \cdot \theta_4 a_1) = \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 0 \\ 0 & 0 & \infty & 0 & \infty \\ \infty & \infty & 0 & \infty & 0 \end{bmatrix}.$$

Algoritam se zaustavlja te dobijamo da je $\theta_4 = \varrho_4$ najveća desno invarijantna matrica kvazi-uređenja na $\mathcal{A}$.

6.4 Najveće slabo desno i slabo levo invarijantne matrice

Ova sekcija posvećena je izračunavanju najvećih slabo desno i slabo levo invarijantnih matrica ekvivalencije i kvazi-uređenja. Prema odgovarajućim definicijama, najpre je potrebno izračunati sve A-kolona vektore kolekcije $\{\tau_u\}_{u \in X^*}$, te je procedura za njihovo računanje formalizovana Algoritmom 12.

Algoritam 12 Računanje svih članova familije $\{\tau_u\}_{u \in X^*}$

Ulaz: TKA $\mathcal{A} = (A, \delta, \sigma, \tau)$ nad alfabetom X i aditivno idempotentnim poluprstenom S

Izlaz: Familija A-kolona vektora $\{\tau_u\}_{u \in X^*}$

1: Inicijalizovati prazan red q čiji su elementi A-kolona vektori, kao i prazno stablo T čiji su čvorovi takođe A-kolona vektori.
2: $\tau_\varepsilon \leftarrow \tau$
3: `Insert`(T, τ_ε)
4: `Enqueue`(q, τ_ε)
5: **while** not `IsEmpty`(q) **do**
6: $\quad \tau_u \leftarrow$ `Dequeue`(q)
7: $\quad$ **for all** $x \in X$ **do**
8: $\quad\quad \tau_{x \cdot u} \leftarrow \delta_x \cdot \tau_u$
9: $\quad\quad$ **if** not `Lookup`$(T, \tau_{x \cdot u})$ **then**
10: $\quad\quad\quad$ `Enqueue`$(q, \tau_{x \cdot u})$
11: $\quad\quad$ **end if**
12: $\quad\quad$ `Insert`$(T, \tau_{x \cdot u}, \tau_u)$
13: $\quad$ **end for**
14: **end while**
15: **return** Unutrašnji čvorovi stabla T (svi različiti članovi familije $\{\tau_u\}_{u \in X^*}$)

Nažalost, kao i u fazi slučaju, ova procedura se ne mora uvek završiti u konačnom broju koraka, s obzirom da kolekcija $\{\tau_u\}_{u \in X^*}$ može biti konačna. Ponovo, kao i u fazi slučaju, ukoliko sa k označimo podpulprsten $S(\delta, \tau)$ od S, tada je vreme izvršenja Algoritma 12 jednako $\mathcal{O}(mnk^{2n})$. Dakle, vreme izvšenja algoritma je eksponencijalno u odnosu na broj stanja od $\mathcal{A}$, jer broj elemenata kolekcije $\{\tau_u\}_{u \in X^*}$ može eksponencijalno da raste. Sa druge strane, broj elemenata ove kolekcije u mnogim

praktičnim primerima je daleko manji od teoretske granice k^n, što omogućuje korišćenje Algoritma 12 u mnogim primerima.

Sada dajemo algoritam za računanje najveće slabo desno invarijantne matrice kvazi-uređenja. Ostale vrste slabo invarijantnih matrica mogu se izračunati na sličan način.

Algoritam 13 Računanje najveće slabo desno invarijantne matrice kvazi-uređenja

Ulaz: TKA $\mathcal{A} = (A, \delta, \sigma, \tau)$ nad alfabetom X i aditivno idempotentnim poluprstenom S

Izlaz: Najveća slabo desno invarijantna matrica kvazi-uređenja

1: Izračunati familiju A-kolona vektora $\{\tau_u\}_{u\in X^*}$ preko Algoritma 12
2: $\varrho \leftarrow \bigodot_{u\in X^*} \tau_u / \tau_u$
3: **return** ϱ

Ponovo, ukoliko podpoluprsten $S(\delta, \tau)$ od S ima k elemenata, tada kolekcija $\{\tau_u\}_{u\in X^*}$ može imati najviše k^n različitih elemenata, Tada računanje Boolovog levog reziduala τ_u / τ_u izvršava se u $\mathcal{O}(n^2 c_+)$ vremenu, računanje kolekcije $\{\tau_u / \tau_u\}_{u\in X^*}$ zahteva $\mathcal{O}(k^n n^2 c_+)$ vreme, i računanje Hadamardovog proizvoda svih matrica iz kolekcije zahteva $\mathcal{O}(k^n n^2 c_+)$ vreme. Dakle, Korak 2 izvršava se u $\mathcal{O}(k^n n^2 c_+)$ vremenu, što ne prekoračuje vreme potrošeno u Koraku 1. Dakle, vreme izvršenja Algoritma 13 je $\mathcal{O}(mnk^{2n})$, isto kao i za Algoritam 12.

6.5 Determinizacija težinskih automata

Neka je A neprazan konačan skup i S proizvoljan poluprsten bez delioca nule. Uređen par $D = (f, g)$ preslikavanja $f : S^A \to S^A$ i $g : S^A \to S$ naziva se *faktorizacija dimenzije* A, ili jednostavno samo *faktorizacija*, ukoliko je:

$$\begin{aligned} &\mu = g(\mu) \cdot f(\mu), \quad \text{za svako } \mu \in S^A, \\ &g(0_A) = 1. \end{aligned}$$

Za svako $\mu \in S^A$ imamo da je $g(\mu) \neq 0$, kao i $\mu = 0_A$ akko $f(\mu) = 0_A$. Faktorizacija $D_N = (f_N, g_N)$, gde je $g_N(\mu) = 1$ i $f_N(\mu) = \mu$, za svako $\mu \in S^A$, naziva se *trivijalna faktorizacija*. U svakom poluprstena bez delioca nule postoji faktorizacija, i to je upravo trivijalna faktorizacija. Faktorizacija $D = (f, g)$ naziva se *maksimalna* ako je $f(a \cdot \mu) = f(\mu)$ za svako $a \in S$ i $\mu \in S^A$ tako da je $a \cdot \mu \neq 0_A$. Kao što je napomenuto u [116], uslov "$a \cdot \mu \neq 0_A$" je neophodan, s obzirom da u suprotnom imamo da je $f(\mu) = f(0 \cdot \mu) = f(0_A)$, za svako $\mu \in S^A$, te pojam maksimalne faktorizacije postaje besmislen. Ako je $D = (f, g)$ maksimalna faktorizacija, tada je $f(f(\mu)) = f(g(\mu) \cdot f(\mu)) = f(\mu)$ za svako $\mu \in S^A$.

Primer 6.12. U ovom primeru dajemo maksimalne faktorizacije za neke aditivno idemptoentne poluprstene. Neka je A neprazan konačan skup.

1. Neka je $(\{0, 1\}, \max, \min, 0, 1)$ Boolov poluprsten. Bilo koji dvoelementni aditivno idempotentni poluprsten izomorfan je Boolovom poluprstenu. U Boolovom poluprstenu moguće je konstruisati jedino trivijalnu faktorizaciju, tj. $g(\mu) = 1$ i $f(\mu) = \mu$, za svako $\mu \in \{0, 1\}^A$, koja je ujedno i maksimalna faktorizacija.

2. U tropskom poluprstenu $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$, definišimo *Mohrievu faktorizaciju* sa $g(\mu) = \min_{a \in A} \mu(a)$ i $f(\mu) = \mu'$, pri čemu je $\mu' \in (\mathbb{R}_+ \cup \{\infty\})^A$ definisano sa $g(\mu) + \mu' = \mu$, za svako $\mu \in (\mathbb{R}_+ \cup \{\infty\})^A$. A-vektor μ' jedinstveno je određen, i Mohrieva faktorizacija je maksimalna faktorizacija.

3. U Viterbievom (probabilističkom) poluprstenu $([0,1], \max, \cdot, 0, 1)$, takođe definišemo *Mohrievu faktorizaciju* sa $g(\mu) = \max_{a \in A} \mu(a)$ i $f(\mu) = \mu'$, pri čemu je $\mu' \in [0,1]^A$ definisano sa $g(\mu) \cdot \mu' = \mu$, za svako $\mu \in [0,1]^A$. A-vektor μ' jedinstveno je određen, i Mohrieva faktorizacija je maksimalna faktorizacija.

Napomenimo da su svi prethodni primeri ujedno i primeri komutativnih poluprstena bez delioca nule.

Sada smo u poziciji da primenimo metode za determinizaciju TKA analogne onima za determinizaciju FKA prikazanih u prethodnim Glavama. Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad pouprstenom S bez delioca nule, i neka je $D = (f, g)$ faktorizacija dimenzije A. Za svako $u \in X^*$ Za svako $u \in X^*$ definišimo A-vektor $\sigma_u^D \in S^A$ induktivno sa: $\sigma_\varepsilon^D = f(\sigma)$ i za svako $u \in X^*$ i $x \in X$ stavimo da je $\sigma_{ux}^D = f(\sigma_u^D \cdot \delta_x)$. Neka je $A^D = \{\sigma_u^D | u \in X^*\}$, i definišimo funkcije $\delta^D : A^D \times X \times A^D \to S$ i $\tau^D : A^D \to S$ saL

$$\delta^D(\mu, x, \nu) = \begin{cases} g(\sigma_u^D \cdot \delta_x), & \mu = \sigma_u^D \text{ i } \nu = \sigma_{ux}^D \\ 0, & \text{inače} \end{cases}, \text{ za svako } u \in X^* \text{ i } x \in X,$$

$$\tau^D(\mu) = \mu \cdot \tau, \text{ za svako } \alpha \in A^D.$$

Naredne dve Teoreme dokazuju se analogno kao i u slučaju fazi automata, a mogu sa naći i u radu Kirsten i Mäurer [116].

Teorema 6.13. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ TKA nad poluprstenom bez delioca nule, i $D = (f, g)$ faktorizacija od A. Tada je $\mathcal{A}^D = (A^D, \{\sigma_\varepsilon^D / g(\sigma)\}, \delta^D, \tau^D)$ dobro definisan KDTA ekvivalentan sa $\mathcal{A}$.*

Teorema 6.14. *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ TKA nad poluprstenom bez delioca nule, $N = (f_N, g_N)$, $M = (f_M, g_M)$ i $D = (f, g)$ trivijalna, maksimalna i proizvoljna faktorizacija dimenzije A, respektivno, kao i $\mathcal{A}^N$, $\mathcal{A}^M$ i $\mathcal{A}^D$ odgovarajući KDTA-i.Tada je $|\mathcal{A}^M| \leq |\mathcal{A}^D|$ i $|\mathcal{A}^M| \leq |\mathcal{A}^N|$.*

Definicija 6.15 (Svojstvo blizanaca)**.** [154] Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ TKA nad poluprstenom. Tada $\mathcal{A}$ zadovoljava *svojstvo blizanaca* ako za sve reči $u, v \in X^*$ kao i za sva stanja $a, b \in A$ važi da, ako je $\sigma_u(a) \neq 0$ i $\sigma_u(b) \neq 0$ i ako postoje dva ciklusa θ_v i η_v u grafu prelaza od $\mathcal{A}$ tako da je $a \in \theta_v$ i $b \in \eta_v$, tada sledi $w(\theta_v) = w(\eta_v)$.

Definicija 6.16 (Min-poluprsten)**.** [116] Za poluprsten S kaže se da je *min-poluprsten* ako zadovoljava jedan od sledeća tri ekvivalentna uslova:

(1) Za svako $x, y \in S$ važi $x + y \in \{x, y\}$;
(2) Postoji linearno uređenje $\leq$ na S tako da je $+ = \min$ (minimum u odnosu na uređenje $\leq$);
(3) Postoji linearno uređenje $\leq$ na S tako da je:
   (a) $+ = \min$ (minimum u odnosu na uređenje $\leq$);
   (b) $\leq$ je kompatibilno u odnosu na $\cdot$.

Svaki min-semiring je aditivno idempotentan, ali obrnuto ne mora da važi - na primer, poluprsten $(2^A, \cup, \cap, \emptyset, A)$ za neprazan skup A.

Teorema 6.17. [116] *Neka je $\mathcal{A} = (A, \sigma, \delta, \tau)$ TKA nad komutativnim, aditivno idempotentnim poluprstenom bez delioca nule, i neka je $D = (f, g)$ maksimalna faktorizacija. Ukoliko $\mathcal{A}$ zadovoljava svojstvo blizanaca, tada je $\mathcal{A}^D$ konačni KDTA.*

Neka je S aditivno idempotentni poluprsten bez delioca nule, i neka je $\varrho \in 2^{A\times A}$ Boolova $A \times A$-matrica. Definišimo TKA $\mathcal{A}_\varrho = (A, \delta_\varrho, \sigma_\varrho, \tau_\varrho)$ sa:

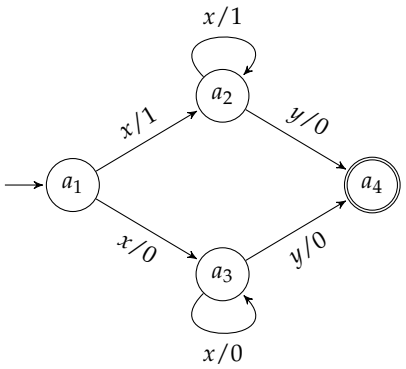
$$\sigma_\varrho = \sigma \cdot \varrho, \quad \tau_\varrho = \tau,$$
$$\delta_\varrho(a, x, b) = (\delta_x \cdot \varrho)(a, b), \quad \text{za svako } x \in X \text{ i } a, b, \in A.$$

Teorema 6.18. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad aditivno idempotentnim poluprstenom bez delioca nule, i $\varrho \in 2^{A\times A}$ refleksivna desno invarijantna matrica, i neka je $\mathcal{A}_\varrho = (A, \delta_\varrho, \sigma_\varrho, \tau_\varrho)$. Tada je $[\![\mathcal{A}]\!] = [\![\mathcal{A}^D_\varrho]\!]$.*

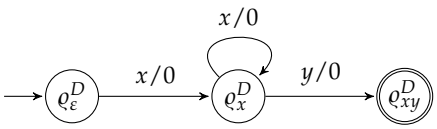
Teorema 6.19. *Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad komutativnim, aditivno idempotentnim poluprstenom bez delioca nule, $D = (f, g)$ maksimalna faktorizacija dimenzije A, i neka su $\varphi, \phi \in 2^{A\times A}$ desno invarijantne matrice kvazi-uređenja. Ako je $\varphi \leq \phi$, tada je $|\mathcal{A}^D_\phi| \leq |\mathcal{A}^D_\varphi|$.*

Ukoliko je poluprsten S i min-poluprsten (i pritom zadovoljava sve uslove iz Teoreme 6.19), tada na osnovu Teoreme 6.17 zaključujemo da se naš determinizacioni metod završava u konačnom broju koraka ako $\mathcal{A}_\varrho$ zadovoljava svojstvo blizanaca.

Naredni primer prikazuje slučaj kada je $\mathcal{A}^D$ beskonačan, ali je $\mathcal{A}^D_\varrho$ konačan KDTA, pri čemu je ϱ najveća desno invarijantna matrica kvazi-uređenja, a D maksimalna faktorizacija dimenzije A.



SLIKA 6.2: Graf prelaza za TKA $\mathcal{A}$ iz Primera 6.20.



SLIKA 6.3: Graf prelaza KDFA $\mathcal{A}^D_\varrho$ ekvivalentnog sa $\mathcal{A}$.

Primer 6.20. Neka je $\mathcal{A} = (A, \delta, \sigma, \tau)$ TKA nad alfabetom $X = \{x, y\}$ i tropskim poluprstenom čiji je graf prelaza dat na Slici 6.2. Može se proveriti da $\mathcal{A}$ ne zadovoljava svojstvo blizanaca, te je stoga $\mathcal{A}^D$ beskonačni KDTA. Sa druge strane, primenom Algoritma 11 dobijamo da je najveća desno invarijantna matrica kvazi-uređenja $\varrho \in 2^{A\times A}$ zadata sa:

$$\varrho = \begin{bmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ 0 & 0 & 0 & \infty \\ \infty & \infty & \infty & 0 \end{bmatrix}.$$

Sada možemo dobiti TKA $\mathcal{A}_\varrho$ koji zadovoljava svojstvo blizanaca. Graf prelaza $\mathcal{A}^D_\varrho$ prikazan je na Slici 6.3.

Konstrukcija dečjeg težinskog automata, kao i težinskog automata tipa Brzozvski analogna je kao u slučaju fazi automata, te upućujemo čitaoca na prethodne Glave za više detalja.

Literatura

[1] L. Aceto, A. Ingolfsdottir, and J. Srba. "The algorithmics of bisimilarity". In: *Advanced Topics in Bisimulation and Coinduction*. Ed. by D. Sangiorgi and J. Rutten. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2011, pp. 100–172.

[2] C. Allauzen and M. Mohri. "Efficient algorithms for testing twins property". In: *Journal of Automata, Languages and Combinatorics* 8.2 (2003), pp. 117–144.

[3] M. Almeida, N. Moreira, and R. Reis. "Finite Automata Minimization". In: *Handbook of Finite State Based Models and Applications*. Ed. by J. Wang. CRC Press/Taylor & Francis Group, 2013, pp. 145–170.

[4] M. Almeida, N. Moreira, and R. Reis. "On the performance of automata minimization algorithms". In: *Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms*. 2007, pp. 3–14.

[5] R. Alur, T. A. Henzinger, and O. Kupferman. "Alternating-time Temporal Logic". In: *Journal of the ACM* 49.5 (2002), pp. 672–713.

[6] B. Aminof, O. Kupferman, and R. Lampert. "Rigorous approximated determinization of weighted automata". In: *Theoretical Computer Science* 480 (2013), pp. 104 –117.

[7] K. Asai and S. Kitajima. "A method for optimizing control of multimodal systems using fuzzy automata". In: *Information Sciences* 3 (1971), pp. 343–353.

[8] J. J. Astrain, J. R. G. de Mendívil, and J. R. Garitagoitia. "Fuzzy automata with ϵ-moves compute fuzzy measures between strings". In: *Fuzzy Sets and Systems* 157.11 (2006), pp. 1550–1559.

[9] J. J. Astrain, J. R. Garitagoitia, J. R. G. de Mendívil, J. Villadangos, and F. Fariña. "Approximate string matching using deformed fuzzy automata: A learning experience". In: *Fuzzy Optimization and Decision Making* 3.2 (2004), pp. 141–155.

[10] J. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Wiley, 1992.

[11] B. De Baets and R. Mesiar. "$\mathcal{F}$-partitions". In: *Fuzzy Sets and Systems* 97 (1998), pp. 211–223.

[12] C. Baier, B. Engelen, and M. Majster-Cederbaum. "Deciding Bisimilarity and Similarity for Probabilistic Processes". In: *Journal of Computer and System Sciences* 60.1 (2000), pp. 187 –231.

[13] M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch. "Squaring transducers: an efficient procedure for deciding functionality and sequentiality". In: *Theoretical Computer Science* 292.1 (2003). Selected Papers in honor of Jean Berstel, pp. 45 –63.

[14] R. Bělohlávek. "Determinism and fuzzy automata". In: *Information Sciences* 143 (2002), pp. 205–209.

[15] R. Bělohlávek. *Fuzzy Relational Systems: Foundations and Principles*. New York: Kluwer, 2002.

[16] R. Bělohlávek and V. Vychodil. *Fuzzy Equational Logic, Studies in Fuzziness and Soft Computing*. Berlin-Heidelberg: Springer, 2005.

[17] J. van Benthem. "Modal correspondence theory". PhD thesis. Universiteit van Amsterdam, Instituut voor Logica en Grondslagenonderzo ek van Exacte Wetenschappen, 1976.

[18] J. Berstel. "Transductions and Context-Free Languages". In: (1979).

[19] J. C. Bezdek and J. D. Harris. "Fuzzy partitions and relations; an axiomatic basis for clustering". In: *Fuzzy Sets and Systems* 1.2 (1978), pp. 111 –127.

[20] G. Birkhoff. *Lattice Theory*. third ed. Providence: American Mathematical Society, 1973.

[21] G. Birkhoff and T. Barti. *Modern applied algebra*. Mc Graw-Hill, inc, 1970.

[22] A. Blanco, M. Delgado, and M.C. Pegalajar. "Fuzzy automaton induction using neural network". In: *International Journal of Approximate Reasoning* 27.1 (2001), pp. 1 –26.

[23] T. S. Blyth. *Lattices and Ordered Algebraic Structures*. Universitext. Springer London, 2005.

[24] U. Bodenhofer, M. De Cock, and E. E. Kerre. "Openings and closures of fuzzy preorderings: theoretical basis and applications to fuzzy rule-based systems". In: *International Journal of General Systems* 32 (2003), pp. 343–360.

[25] F. Bonchi, M. Bonsangue, M. Boreale, J. Rutten, and A. Silva. "A coalgebraic perspective on linear weighted automata". In: *Information and Computation* 211 (2012), pp. 77 –105.

[26] S. Bozapalidis and O. Louscou-Bozapalidou. "On the recognizability of fuzzy languages I". In: *Fuzzy Sets and Systems* 157 (2006), pp. 2394–2402.

[27] S. Bozapalidis and O. Louscou-Bozapalidou. "On the recognizability of fuzzy languages II". In: *Fuzzy Sets and Systems* 159 (2008), pp. 107–113.

[28] J. S. de Bruin, H. Steltzer, A. Rappelsberger, and K.-P. Adlassnig. "Creating Clinical Fuzzy Automata with Fuzzy Arden Syntax". In: *AMIA ... Annual Symposium proceedings. AMIA Symposium* 2017 (2017), pp. 475–484.

[29] J. A. Brzozowski. "Canonical regular expressions and minimal state graphs for definite events". In: *Mathematical theory of Automata*. Volume 12 of MRI Symposia Series. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1962, pp. 529–561.

[30] P. Buchholz. "Bisimulation relations for weighted automata". In: *Theoretical Computer Science* 393.1 (2008), pp. 109 –123.

[31] A. L. Buchsbaum, R. Giancarlo, and J. R. Westbrook. "On the determinization of weighted finite automata". In: *SIAM Journal on Computing* 30.5 (2000), pp. 1502–1531.

[32] A. L. Buchsbaum, R. Giancarloand, and J. R. Westbrook. "An Approximate Determinization Algorithm for Weighted Finite-State Automata". In: *Algorithmica* 30.4 (2001), pp. 503–526.

[33] C. S. Calude, E. Calude, and B. Khoussainov. "Finite nondeterministic automata: Simulation and minimality". In: *Theoretical Computer Science* 242.1 (2000), pp. 219 –235.

[34] C. Câmpeanu, N. Sântean, and S. Yu. "Mergible states in large NFA". In: *Theoretical Computer Science* 330.1 (2005), pp. 23 –34.

[35] J.-M. Champarnaud and F. Coulon. "NFA Reduction Algorithms by Means of Regular Inequalities". In: *Developments in Language Theory*. Ed. by Z .Ésik and Z. Fülöp. Berlin, Heidelberg: Springer, 2003, pp. 194–205.

[36] J.-M. Champarnaud and F. Coulon. "NFA Reduction Algorithms by Means of Regular Inequalities". In: *Theoretical Computer Science* 327.3 (2004), pp. 241–253. Erratum-ibid. Theoretical Computer Science 347 (2005) 437–440.

[37] J.-M. Champarnaud, A. Khorsi, and T. Paranthoën. "Split and join for minimizing: Brzozowski's algorithm". In: *Proceedings of the Prague Stringology Conference*. Ed. by M. Balík and M. Simánek. 2002, pp. 96–104.

[38] M. Ćirić and J. Ignjatović. *Teorija algoritama, automata i jezika - zbirka zadataka*. Univerzitet u Nišu, Prirodno - matematički fakultet, 2012.

[39] M. Ćirić, J. Ignjatović, and S. Bogdanović. "Fuzzy equivalence relations and their equivalence classes". In: *Fuzzy Sets and Systems* 158 (2007), pp. 1295 – 1313.

[40] M. Ćirić, J. Ignjatović, and S. Bogdanović. "Uniform fuzzy relations and fuzzy functions". In: *Fuzzy Sets and Systems* 160.8 (2009), pp. 1054 –1081.

[41] M. Ćirić, J. Ignjatović, N. Damljanović, and M. Bašić. "Bisimulations for fuzzy automata". In: *Fuzzy Sets and Systems* 186.1 (2012), pp. 100 –139.

[42] M. Ćirić, J. Ignjatović, I. Jančić, and N. Damljanović. "Computation of the greatest simulations and bisimulations between fuzzy automata". In: *Fuzzy Sets and Systems* 208 (2012), pp. 22 –42.

[43] M. Ćirić, M. Droste, J. Ignjatović, and H. Vogler. "Determinization of weighted finite automata over strong bimonoids". In: *Information Sciences* 180 (2010), pp. 3497–3520.

[44] M. Ćirić, A. Stamenković, J. Ignjatović, and T. Petković. "Factorization of Fuzzy Automata". In: *Proceedings of the 16th International Conference on Fundamentals of Computation Theory*. FCT'07. Budapest, Hungary: Springer-Verlag, 2007, pp. 213–225.

[45] M. Ćirić, A. Stamenković, J. Ignjatović, and T. Petković. "Fuzzy relation equations and reduction of fuzzy automata". In: *Journal of Computer and System Sciences* 76.7 (2010), pp. 609 –633.

[46] M. Ćirić, J. Ignjatović, M. Bašić, and I. Jančić. "Nondeterministic automata: Equivalence, bisimulations, and uniform relations". In: *Information Sciences* 261 (2014), pp. 185 –218.

[47] A. Clark. *Elements of Abstract Algebra*. Dover Books on Mathematics Series. Dover Publications, 1984.

[48] R. Cleaveland and O. Sokolsky. "CHAPTER 6 - Equivalence and Preorder Checking for Finite-State Systems". In: *Handbook of Process Algebra*. Ed. by J.A. Bergstra, A. Ponse, and S.A. Smolka. Amsterdam: Elsevier Science, 2001, pp. 391 –424.

[49] M. De Cock, U. Bodenhofer, and E. E. Kerre. "Modelling Linguistic Expressions Using Fuzzy Relations". In: *Proceedings of the 6th International Conference on Soft Computing*. 2000, pp. 353–360.

[50] C. Courcoubetis and M. Yannakakis. “The Complexity of Probabilistic Verification”. In: *J. ACM* 42.4 (1995), pp. 857–907.

[51] N. Damljanović, M. Ćirić, and J. Ignjatović. “Bisimulations for weighted automata over an additively idempotent semiring”. In: *Theoretical Computer Science* 534 (2014), pp. 86 –100.

[52] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. 2nd ed. Cambridge University Press, 2002.

[53] M. Demirci. “On many-valued partitions and many valued equivalence relations”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 11 (2003), pp. 235–253.

[54] M. Demirci. “Topological properties of the class of generators of an indistinguishability operator”. In: *Fuzzy Sets and Systems* 143 (2004), pp. 413–426.

[55] M. Demirci and J. Recasens. “Fuzzy groups, fuzzy functions and fuzzy equivalence relations”. In: *Fuzzy Sets and Systems* 144.3 (2004), pp. 441 –458.

[56] A. Dovier, C. Piazza, and A. Policriti. “A Fast Bisimulation Algorithm”. In: *Computer Aided Verification*. Ed. by G. Berry, H. Comon, and A. Finkel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 79–90.

[57] A. Dovier, C. Piazza, and A. Policriti. “An efficient algorithm for computing bisimulation equivalence”. In: *Theoretical Computer Science* 311.1 (2004), pp. 221 –256.

[58] M. Droste and P. Gastin. “On Aperiodic and Star-free Formal Power Series in Partially Commuting Variables”. In: *Formal Power Series and Algebraic Combinatorics*. Ed. by D. Krob, A. A. Mikhalev, and A. V. Mikhalev. Berlin, Heidelberg: Springer, 2000, pp. 158–169.

[59] M. Droste and W. Kuich. “Weighted finite automata over hemirings”. In: *Theoretical Computer Science* 485 (2013), pp. 38–48.

[60] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. An EATCS Series. Springer Publishing Company, Incorporated, 2009.

[61] M. Droste, T. Stüber, and H. Vogler. “Weighted finite automata over strong bimonoids”. In: *Information Sciences* 180 (2010), pp. 156–166.

[62] M. Droste and H. Vogler. “Weighted automata and multi-valued logics over arbitrary bounded lattices”. In: *Theoretical Computer Science* 418 (2012), pp. 14 –36.

[63] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, Inc., 1980.

[64] S. Eilenberg. *Automata, languages, and machines*. Academic press, 1974.

[65] J. Elorza and P. Burillo. “Connecting fuzzy preorders, fuzzy consequence operators and fuzzy closure and co-closure systems”. In: *Fuzzy Sets and Systems* 139 (2003), pp. 601–613.

[66] F. Esteva, L. Godo, and À. García-Cerdaña. “On the Hierarchy of t-norm Based Residuated Fuzzy Logics”. In: *Beyond Two: Theory and Applications of Multiple-Valued Logic*. Ed. by M. Fitting and E. Orłowska. Heidelberg: Physica-Verlag HD, 2003, pp. 251–272.

[67] J. C. Fodor and M. Roubens. “Structure of transitive valued binary relations”. In: *Mathematical Social Sciences* 30 (1995), pp. 71–94.

[68] M. Forti and F. Honsell. “Set theory with free construction principles”. In: *Annali della Scuola Normale Superiore di Pisa - Classe di Scienze* Ser. 4, 10.3 (1983), pp. 493–522.

[69] J. Friedl. *Mastering Regular Expressions*. 3rd edn. Sebastopol, CA: O’Reilly Media, 2006.

[70] J. R. Garitagoitia, J. R. G. de Mendívil, J. Echanobe, J. J. Astrain, and F. Fariña. “Deformed fuzzy automata for correcting imperfect strings of fuzzy symbols”. In: *IEEE Transactions on Fuzzy Systems* 11.3 (2003), pp. 299–310.

[71] S. Gaubert. “Performance evaluation of (max,+) automata”. In: *IEEE Transactions on Automatic Control* 40.12 (1995), pp. 2014–2025.

[72] S. Gaubert and J. Mairesse. “Modeling and analysis of timed Petri nets using heaps of pieces”. In: *IEEE Transactions on Automatic Control* 44.4 (1999), pp. 683–697.

[73] R. Gentilini, C. Piazza, and A. Policriti. “From Bisimulation to Simulation: Coarsest Partition Problems”. In: *Journal of Automated Reasoning* 31.1 (2003), pp. 73–103.

[74] R. van Glabbeek and B. Ploeger. *Five determinization algorithms*. Tech. rep. CS-Report 08-14, Eindhoven University of Technology, 2008.

[75] R. van Glabbeek and B. Ploeger. “Five determinization algorithms”. In: *in: O.H. Ibarra, B. Ravikumar (Eds.), CIAA 2008, Lecture Notes in Computer Science* 5148 (2008), pp. 161–170.

[76] J. S. Golan. *The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science*. Essex, UK, UK: Addison-Wesley Longman Ltd., 1992.

[77] M. Gondran and M. Minoux. “Dioïds and semirings: Links to fuzzy sets and other applications”. In: *Fuzzy Sets and Systems* 158.12 (2007), pp. 1273 –1294.

[78] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings*. Springer, 2008.

[79] S. Gottwald. *A Treatise on Many-Valued Logics*. Baldock: Research Studies Press, 2001.

[80] G. Grätzer. *Lattice Theory: Foundation*. Birkhäuser Basel, 2011.

[81] G. Grätzer. *Universal algebra*. University series in higher mathematics. Van Nostrand, 1968.

[82] D. Gries. “Describing an algorithm by Hopcroft”. In: *Acta Informatica* 2.2 (1973), pp. 97–109.

[83] M. M. Gupta, G. N. Saridis, and B. R. Gaines. *Fuzzy Automata and Decision Processes*. New York: North-Holland, 1977.

[84] M. Habib, C. Paul, and L. Viennoti. “A synthesis on partition refinement: A useful routine for strings, graphs, boolean matrices and automata”. In: *STACS 98*. Ed. by M. Morvan, C. Meinel, and D. Krob. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 25–38.

[85] U. Hafner. “Low bit rate image and video coding with weighted finite automata”. In: *ICIP*. 1997.

[86] P. Hájek. *Metamathematics of Fuzzy Logic*. Dordrecht: Kluwer Academic Publishers, 1998.

[87] K. Hashiguchi. “Improved Limitedness Theorems on Finite Automata with Distance Functions”. In: *Theoretical Computer Science* (1990), pp. 27–38.

[88] U. Hebisch and H. J. Weinert. *Semirings: Algebraic Theory And Applications In Computer Science*. Series In Algebra. World Scientific Publishing Company, 1998.

[89] U. Höhle. "Commutative, residuated 1—monoids". In: *Non-Classical Logics and their Applications to Fuzzy Subsets: A Handbook of the Mathematical Foundations of Fuzzy Set Theory*. Ed. by U. Höhle and E. P. Klement. Dordrecht: Springer Netherlands, 1995, pp. 53–106.

[90] U. Höhle. "On the Fundamentals of Fuzzy Set Theory". In: *Journal of Mathematical Analysis and Applications* 201 (1996), pp. 786–826.

[91] J. E. Hopcroft. "An $n \log n$ algorithm for minimizing states in a finite automaton". In: *Theory of Machines and Computations*. Ed. by Z. Kohavi and A. Paz. Academic Press, 1971, pp. 189 –196.

[92] J. E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, third ed.* Addison-Wesley, 2007.

[93] D. A. Huffman. "The Synthesis of Sequential Switching Circuits". In: *Journal of Symbolic Logic* 20.1 (1955), pp. 69–70.

[94] J. Ignjatović. "Fazi relacije, automati i jezici". PhD thesis. University of Niš, Faculty of Sciences and Mathematics, 2007.

[95] J. Ignjatović and M. Ćirić. *Automati i formalni jezici*. Univerzitet u Nišu, Prirodno - matematički fakultet, 2016.

[96] J. Ignjatović and M. Ćirić. "Weakly linear systems of fuzzy relation inequalities and their applications: A brief survey". In: *Filomat* 26.2 (2012), pp. 207–241.

[97] J. Ignjatović, M. Ćirić, and S. Bogdanović. "Determinization of fuzzy automata with membership values in complete residuated lattices". In: *Information Sciences* 178 (2008), pp. 164–180.

[98] J. Ignjatović, M. Ćirić, and S. Bogdanović. "On the greatest solutions to weakly linear systems of fuzzy relation inequalities and equations". In: *Fuzzy Sets and Systems* 161.24 (2010), pp. 3081–3113.

[99] J. Ignjatović, M. Ćirić, B. Šešelja, and A. Tepavčević. "Fuzzy relational inequalities and equations, fuzzy quasi-orders, closures and openings of fuzzy sets". In: *Fuzzy Sets and Systems* 260 (2015), pp. 1–24.

[100] J. Ignjatović, M. Ćirić, S. Bogdanović, and T. Petković. "Myhill-Nerode type theory for fuzzy languages and automata". In: *Fuzzy Sets and Systems* 161 (2010), pp. 1288–1324.

[101] J. Ignjatović, M. Ćirić, N. Damljanović, and I. Jančić. "Weakly linear systems of fuzzy relation inequalities: The heterogeneous case". In: *Fuzzy Sets and Systems* 199 (2012), pp. 64–91.

[102] K. Culik II and J. Kari. "Image compression using weighted finite automata". In: *Computers & Graphics* 17.3 (1993), pp. 305 –313.

[103] L. Ilie, G. Navarro, and S. Yu. "On NFA Reductions". In: *Theory Is Forever: Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday*. Ed. by J. Karhumäki, H. Maurer, G. Păun, and G. Rozenber. Berlin, Heidelberg: Springer, 2004, pp. 112–124.

[104] L. Ilie, R. Solis-Oba, and S. Yu. "Reducing the Size of NFAs by Using Equivalences and Preorders". In: *Combinatorial Pattern Matching*. Ed. by A. Apostolico, M. Crochemore, and K. Park. Berlin, Heidelberg: Springer, 2005, pp. 310–321.

[105] L. Ilie and S. Yu. "Algorithms for Computing Small NFAs". In: *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*. MFCS '02. London, UK, UK: Springer-Verlag, 2002, pp. 328–340.

[106] L. Ilie and S. Yu. "Reducing NFAs by invariant equivalences". In: *Theoretical Computer Science* 306.1 (2003), pp. 373 –390.

[107] I. Jančić. "Weak bisimulations for fuzzy automata". In: *Fuzzy Sets and Systems* 249 (2014), pp. 49–72.

[108] Z. Jančić and M. Ćirić. "Brzozowski type determinization for fuzzy automata". In: *Fuzzy Sets and Systems* 249 (2014), pp. 73–82.

[109] Z. Jančić, J. Ignjatović, and M. Ćirić. "An improved algorithm for determinization of weighted and fuzzy automata". In: *Information Sciences* 181 (2011), pp. 1358–1368.

[110] Z. Jančić, I. Micić, J. Ignjatović, and M. Ćirić. "Further improvement of determinization methods for fuzzy finite automata". In: *Fuzzy Sets and Systems* 301 (2015), pp. 79–102.

[111] B. Jayaram and R. Mesiar. "I-Fuzzy equivalence relations and I-fuzzy partitions". In: *Information Sciences* 179.9 (2009), pp. 1278 –1297.

[112] Z. Jiang, B. Litow, and O. de Vel. "Similarity Enrichment in Image Compression through Weighted Finite Automata". In: *Computing and Combinatorics*. Ed. by D.-Z. Du, P. Eades, V. Estivill-Castro, X. Lin, and A. Sharma. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 447–456.

[113] J. Jin, Q. Li, and Y. Li. "Algebraic properties of L-fuzzy finite automata". In: *Information Sciences* 234 (2013), pp. 182–202.

[114] P. C. Kanellakis and S. A. Smolka. "CCS expressions, finite state processes, and three problems of equivalence". In: *Information and Computation* 86.1 (1990), pp. 43 –68.

[115] Frank Katritzke. "Refinements of data compression using weighted finite automata". PhD thesis. Universität Siegen, 2001.

[116] D. Kirsten and L. Mäurer. "On the determinization of weighted automata". In: *Journal of Automata, Languages and Combinatorics* 10 (2005), pp. 287–312.

[117] F. Klawonn. "Fuzzy points, fuzzy relations and fuzzy functions". In: *Discovering World with Fuzzy Logic*. Ed. by V. Novâk and I. Perfilieva. Heidelberg: Physica-Verlag, 2000, pp. 431–453.

[118] F. Klawonn and R. Kruse. "From fuzzy sets to indistinguishability and back". In: *Proc. First ICSC Internat. Symp. on Fuzzy Logic, Zürich, Switzerland*. Ed. by N. Steele. Millet: ICSC Academic Press, 1995, A57–A59.

[119] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Dordrecht: Kluwer Academic Publishers, 2000.

[120] E. P. Klement, R. Mesiar, and E. Pap. "Triangular norms. Position paper I: basic analytical and algebraic properties". In: *Fuzzy Sets and Systems* 143.1 (2004), pp. 5 –26.

[121] E. P. Klement, R. Mesiar, and E. Pap. "Triangular norms. Position paper III: continuous t-norms". In: *Fuzzy Sets and Systems* 145.3 (2004), pp. 439 –454.

[122] I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. "Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton". In: *Theoretical Computer Science* 327.3 (2004), pp. 349–373.

[123] T. Knuutila. "Re-describing an algorithm by Hopcroft". In: *Theoretical Computer Science* 250.1 (2001), pp. 333 –363.

[124] S. Konstantinidis, N. Santean, and S. Yu. "Fuzzification of rational and recognizable sets". In: *Fundamenta Informaticae* 76 (2007), pp. 413–447.

[125] K. Krithivasan and K. Sharda. "Fuzzy ω-automata". In: *Information Sciences* 138 (2001), pp. 257–281.

[126] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Monographs in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2012.

[127] O. Kupferman and Y. Lustig. "Lattice Automata". In: *Verification, Model Checking, and Abstract Interpretation*. Ed. by B. Cook and A. Podelski. Berlin, Heidelberg: Springer, 2007, pp. 199–213.

[128] V. B. Kuzmin. *Building Group Decisions in Spaces of Strict and Fuzzy Binary Relations*. Russian. Moscow: Nauka, 1982.

[129] H. Lai and D. Zhang. "Fuzzy preorder and fuzzy topology". In: *Fuzzy Sets and Systems* 157 (2006), pp. 1865–1885.

[130] M. V. Lawson. *Finite Automata*. Boca Raton, London: Chapman & Hall, CRC Press, 2004.

[131] E. T. Lee and L. A. Zadeh. "Note on fuzzy languages". In: *Information Sciences* 1.4 (1969), pp. 421–434.

[132] H. Lei and Y. M. Li. "Minimization of states in automata theory based on finite lattice-ordered monoids". In: *Information Sciences* 177 (2007), pp. 1413–1421.

[133] P. Li and Y. M. Li. "Algebraic properties of LA-languages". In: *Information Sciences* 176 (2006), pp. 3232–3255.

[134] Y. Li. "Finite automata theory with membership values in lattices". In: *Information Sciences* 181 (2011), pp. 1003–1017.

[135] Y. M. Li and W. Pedrycz. "Fuzzy finite automata and fuzzy regular expressions with membership values in lattice ordered monoids". In: *Fuzzy Sets and Systems* 156 (2005), pp. 68–92.

[136] Y. M. Li and W. Pedrycz. "Minimization of lattice finite automata and its application to the decomposition of lattice languages". In: *Fuzzy Sets and Systems* 158 (2007), pp. 1423–1436.

[137] Z. Li, P. Li, and Y. M. Li. "The relationships among several types of fuzzy automata". In: *Information Sciences* 176 (2006), pp. 2208–2226.

[138] N. Lynch and F. Vaandrager. "Forward and Backward Simulations". In: *Information and Computation* 121.2 (1995), pp. 214 –233.

[139] D. S. Malik, J. N. Mordeson, and M.K. Sen. "Minimization of fuzzy finite automata". In: *Information Sciences* 113 (1999), pp. 323–330.

[140] D. S. Malik, J. N. Mordeson, and M.K. Sen. "On fuzzy regular languages". In: *Information Sciences* 88 (1996), pp. 263–273.

[141] F. Marass and C. Upton. "Sequence Searcher: A Java tool to perform regular expression and fuzzy searches of multiple DNA and protein sequences". In: *BMC Research Notes* 2.1 (2009).

[142] J. R. G. de Mendívil. "A generalization of Myhill-Nerode theorem for fuzzy languages". In: *Fuzzy Sets and Systems* 301 (2016), pp. 103–115.

[143] J. R. G. de Mendívil. "Conditions for Minimal Fuzzy Deterministic Finite Automata via Brzozowski's Procedure". In: *IEEE Transactions on Fuzzy Systems* 26.4 (2018), pp. 2409–2420.

[144] J. R. G. de Mendívil and J. R. Garitagoitia. "Determinization of fuzzy automata via factorization of fuzzy states". In: *Information Sciences* 283 (2014), pp. 165–179.

[145] J. R. G. de Mendívil and J. R. Garitagoitia. "Fuzzy languages with infinite range accepted by fuzzy automata: Pumping Lemma and determinization procedure". In: *Fuzzy Sets and Systems* 249 (2014), pp. 1–26.

[146] I. Micić, Z. Jančić, and S. Stanimirović. "Computation of the greatest right and left invariant fuzzy quasi-orders and fuzzy equivalences". In: *Fuzzy Sets and Systems* 339 (2018), pp. 99 –118.

[147] I. Micić, Z. Jančić, J. Ignjatović, and M. Ćirić. "Determinization of fuzzy automata by means of the degrees of language inclusion". In: *IEEE Transactions on Fuzzy Systems* 23.6 (2015), pp. 2144–2153.

[148] R. Milner. *A Calculus of Communicating Systems*. Berlin, Heidelberg: Springer-Verlag, 1982.

[149] R. Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.

[150] R. Milner. *Communication and concurrency*. Vol. 84. Prentice hall New York etc., 1989.

[151] M. Mizumoto, J. Toyoda, and K. Tanaka. "Some considerations on fuzzy automata". In: *Journal of Computer and System Sciences* 3 (1969), pp. 409–422.

[152] M. Mizumoto, J. Toyoda, and K. Tanaka. "Various kinds of automata with weights". In: *Journal of Computer and System Sciences* 10.2 (1975), pp. 219 –236.

[153] J. Močkoř. "Fuzzy and non-deterministic automata". In: *Soft Computing* 3 (1999), pp. 221–226.

[154] M. Mohri. "Finite-state transducers in language and speech processing". In: *Computational Linguistics* 23.2 (1997), pp. 269–311.

[155] M. Mohri, F. Pereira, and M. Riley. "Weighted finite-state transducers in speech recognition". In: *Computer Speech & Language* 16.1 (2002), pp. 69 –88.

[156] E. F. Moore. "Gedanken-Experiments on Sequential Machines". In: *The Journal of Symbolic Logic* 23.1 (1958), p. 60.

[157] J. N. Mordeson and D.S. Malik. *Fuzzy automata and languages: Theory and Applications*. Boca Raton, London: Chapman & Hall, CRC Press, 2002.

[158] V. Murali. "Fuzzy equivalence relations". In: *Fuzzy Sets and Systems* 30.2 (1989), pp. 155 –163.

[159] S. Ovchinnikov. “Similarity relations, fuzzy partitions, and fuzzy orderings”. In: *Fuzzy Sets and Systems* 40.1 (1991), pp. 107 –126.

[160] R. Paige and R. E. Tarjan. “Three partition refinement algorithms”. In: *SIAM Journal on Computing* 16.6 (1987), pp. 973–989.

[161] D. Park. “Concurrency and automata on infinite sequences”. In: *Theoretical Computer Science*. Ed. by P. Deussen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 167–183.

[162] W. Pedrycz and A. Gracek. “Learning of fuzzy automata”. In: *International Journal of Computational Intelligence and Applications* 1.1 (2001), pp. 19–33.

[163] K. Peeva. “Finite L-fuzzy acceptors, regular L-fuzzy grammars and syntactic pattern recognition”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12 (2004), pp. 89–104.

[164] K. Peeva. “Finite L-fuzzy machines”. In: *Fuzzy Sets and Systems* 141 (2004), pp. 415–437.

[165] K. Peeva and Y. Kyosev. *Fuzzy Relational Calculus: Theory, Applications and Software (with CD-ROM)*. Advances in fuzzy systems. World Scientific, 2004.

[166] K. Peeva and Z. Zahariev. “Computing behavior of finite fuzzy machines – Algorithm and its application to reduction and minimization”. In: *Information Sciences* 178 (2008), pp. 4152–4165.

[167] F. Pereira and M. Riley. “Speech Recognition by Composition of Weighted Finite Automata”. In: *Finite-State Language Processing*. 1997.

[168] F. Pereira, M. Riley, and R. Sproat. “Weighted Rational Transductions and Their Application to Human Language Processing”. In: *Proceedings of the Workshop on Human Language Technology*. HLT ’94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, pp. 262–267.

[169] A. Pnueli and R. Rosner. “On the synthesis of an asynchronous reactive module”. In: *Automata, Languages and Programming*. Ed. by G. Ausiello, M. Dezani-Ciancaglini, and S. R. Della Rocca. Berlin, Heidelberg: Springer, 1989, pp. 652–671.

[170] D. Qiu. “A note on Trillas’ CHC models”. In: *Artificial Intelligence* 171.4 (2007), pp. 239 –254.

[171] D. W. Qiu. “Automata theory based on completed residuated lattice-valued logic (I)”. In: *Science in China, Ser. F* 44.6 (2001), pp. 419–429.

[172] D. W. Qiu. “Automata theory based on completed residuated lattice-valued logic (II)”. In: *Science in China, Ser. F* 45.6 (2002), pp. 442–452.

[173] D. W. Qiu. “Characterizations of fuzzy finite automata”. In: *Fuzzy Sets and Systems* 141 (2014), pp. 391–414.

[174] D. W. Qiu. “Pumping lemma in automata theory based on complete residuated lattice-valued logic: A note”. In: *Fuzzy Sets and Systems* 157 (2006), pp. 2128–2138.

[175] D. W. Qiu. “Supervisory control of fuzzy discrete event systems: a formal approach”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 35.1 (2005), pp. 72–88.

[176] M. O. Rabin and D. Scott. “Finite automata and their decision problems”. In: *IBM Journal of Research and Development* 3.2 (1959), pp. 114–125.

[177] G. Rahonis. “Fuzzy languages”. In: *Handbook of Weighted Automata*. Ed. by M. Droste, W. Kuich, and H. Vogler. Berlin: Springer, 2009.

[178] F. Ranzato and F. Tapparo. “Generalizing the Paige–Tarjan algorithm by abstract interpretation”. In: *Information and Computation* 206.5 (2008), pp. 620 – 651.

[179] G. G. Rigatos. “Fault detection and isolation based on fuzzy automata”. In: *Information Sciences* 179.12 (2009), pp. 1893 –1902.

[180] M. Roggenbach and M. Majster-Cederbaum. “Towards a unified view of bisimulation: a comparative study”. In: *Theoretical Computer Science* 238.1 (2000), pp. 81 –130.

[181] S. Roman. *Lattices and Ordered Sets*. Springer New York, 2008.

[182] J. C. Rosales and P. A. García Sánchez. *Finitely Generated Commutative Monoids*. New York, Commack: Nova Science Publishers, 1999.

[183] J. Sakarovitch. *Elements of Automata Theory*. New York: Cambridge University Press, 2009.

[184] A. Salomaa and M. Soittola. *Automata: Theoretic Aspects of Formal Power Series*. Berlin, Heidelberg: Springer-Verlag, 1978.

[185] D. Sangiorgi. “On the origins of bisimulation, coinduction, and fixed points”. In: *Universita of Bologna, Italy* (2007).

[186] E. S. Santos. “Fuzzy automata and languages”. In: *Information Sciences* 10 (1976), pp. 193–197.

[187] E. S. Santos. “Maximin automata”. In: *Information and Control* 13.4 (1968), pp. 363–377.

[188] E. S. Santos. “Maxproduct machines”. In: *Journal of Mathematical Analysis and Applications* 37 (1972), pp. 677–686.

[189] E. S. Santos. “On reduction of Maximin machines”. In: *Journal of Mathematical Analysis and Applications* 40 (1972), pp. 60–78.

[190] B. Schroeder. *Ordered Sets: An Introduction*. Birkhäuser Boston, 2003.

[191] M. P. Schützenberger. “On the definition of a family of automata”. In: *Information and Control* 4.2 (1961), pp. 245 –270.

[192] B. Schweizer and A. Sklar. “Statistical metric spaces”. In: *Pacific Journal of Mathematics* 10 (1960), pp. 313–334.

[193] M. Shamsizadeh, M. M. Zahedi, and K. Abolpour. “Bisimulation for BL-general fuzzy automata”. In: *Iranian Journal of Fuzzy Systems* 13.4 (2016), pp. 35 –50.

[194] A. Stamenković and M. Ćirić. “Construction of fuzzy automata from fuzzy regular expressions”. In: *Fuzzy Sets and Systems* 199 (2012), pp. 1 –27.

[195] A. Stamenković, M. Ćirić, and M. Bašić. “Ranks of fuzzy matrices. Applications in state reduction of fuzzy automata”. In: *Fuzzy Sets and Systems* 333 (2018), pp. 124 –139.

[196] A. Stamenković, M. Ćirić, and J. Ignjatović. “Different models of automata with fuzzy states”. In: *Facta Universitatis: Series Mathematics and Informatics* 30.3 (2015), pp. 235–253.

[197] A. Stamenković, M. Ćirić, and J. Ignjatović. “Reduction of fuzzy automata by means of fuzzy quasi-orders”. In: *Information Sciences* 275 (2014), pp. 168 –198.

[198] S. Stanimirović, M. Ćirić, and J. Ignjatović. "Determinization of fuzzy automata by factorizations of fuzzy states and right invariant fuzzy quasi-orders". In: *Information Sciences* 469 (2018), pp. 79–100.

[199] S. Stanimirović, A. Stamenković, and M. Ćirić. "Improved algorithms for computing the greatest right and left invariant Boolean matrices and their application". In: *Filomat* (accepted for publication).

[200] D. Tabakov and M. Y. Vardi. "Experimental Evaluation of Classical Automata Constructions". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by G. Sutcliffe and A. Voronkov. Berlin, Heidelberg: Springer, 2005, pp. 396–411.

[201] P. Terry. *Compiling with C# and Java*. Pearson/Addison-Wesley, 2005.

[202] S. P. Tiwari and A. K. Singh. "On minimal realization of fuzzy behaviour and associated categories". In: *Journal of Applied Mathematics and Computing* 45.1 (2014), pp. 223–234.

[203] D. Vaida. "Notes on Partially-Ordered Structures in Computer Science: I. PA-Ordered Semirings and Some Related Structures". In: *Journal of Universal Computer Science* 6.1 (2000), pp. 201–211.

[204] L. Valverde. "On the structure of F-indistinguishability operators". In: *Fuzzy Sets and Systems* 17 (1985), pp. 313–328.

[205] M. Y. Vardi. "Automatic Verification of Probabilistic Concurrent Finite State Programs". In: *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*. SFCS '85. Washington, DC, USA: IEEE Computer Society, 1985, pp. 327–338.

[206] R. Verma and S. P. Tiwari. "Distinguishability and completeness of crisp deterministic fuzzy automata". In: *Iranian Journal of Fuzzy Systems* 14.5 (2017), pp. 19–30.

[207] B. W. Watson. "Taxonomies and toolkit of regular languages algorithms". PhD thesis. Eindhoven University of Technology, 1995.

[208] W. Wechler. *The Concept of Fuzziness in Automata and Language Theory*. Berlin: T, Akademie-Verlag, 1978.

[209] W. G. Wee. "On generalizations of adaptive algorithm and application of the fuzzy sets concept to pattern classification". PhD thesis. Purdue University, 1967.

[210] W. G. Wee and K. S. Fu. "A formulation of fuzzy automata and its application as a model of learning systems". In: *IEEE Transactions on Systems, Man, and Cybernetics* 5.3 (1969), pp. 215–223.

[211] H. Wu and Y. Deng. "Logical characterizations of simulation and bisimulation for fuzzy transition systems". In: *Fuzzy Sets and Systems* 301 (2016), pp. 19 –36.

[212] H. Wu, T. Chen, T. Han, and Y. Chen. "Bisimulations for fuzzy transition systems revisited". In: *International Journal of Approximate Reasoning* 99 (2018), pp. 1 –11.

[213] L. H. Wu and D. W. Qiu. "Automata theory based on complete residuated lattice-valued logic: reduction and minimization". In: *Fuzzy Sets and Systems* 161.12 (2010), pp. 1635–1656.

[214] H. Xing and D. W. Qiu. "Automata theory based on complete residuated lattice-valued logic: A categorical approach". In: *Fuzzy Sets and Systems* 160 (2009), pp. 2416–2428.

[215] H. Xing and D. W. Qiu. "Pumping lemma in context-free grammar theory based on complete residuated lattice-valued logic". In: *Fuzzy Sets and Systems* 160 (2009), pp. 1141–1151.

[216] H. Xing, D. W. Qiu, and F. C. Liu. "Automata theory based on complete residuated lattice-valued logic: Pushdown automata". In: *Fuzzy Sets and Systems* 160 (2009), pp. 1125–1140.

[217] H. Xing, D. W. Qiu, F. C. Liu, and Z. J. Fan. "Equivalence in automata theory based on complete residuated lattice-valued logic". In: *Fuzzy Sets and Systems* 158 (2007), pp. 1407–1422.

[218] M. S. Ying. "A formal model of computing with words". In: *IEEE Transactions on Fuzzy Systems* 10.5 (2002), pp. 640–652.

[219] L. A. Zadeh. "Fuzzy sets". In: *Information and Control* 8 (1965), pp. 338–353.

[220] L. A. Zadeh. "Similarity relations and fuzzy orderings". In: *Information Sciences* 3.2 (1971), pp. 177 –200.

Indeks

Dodatak A

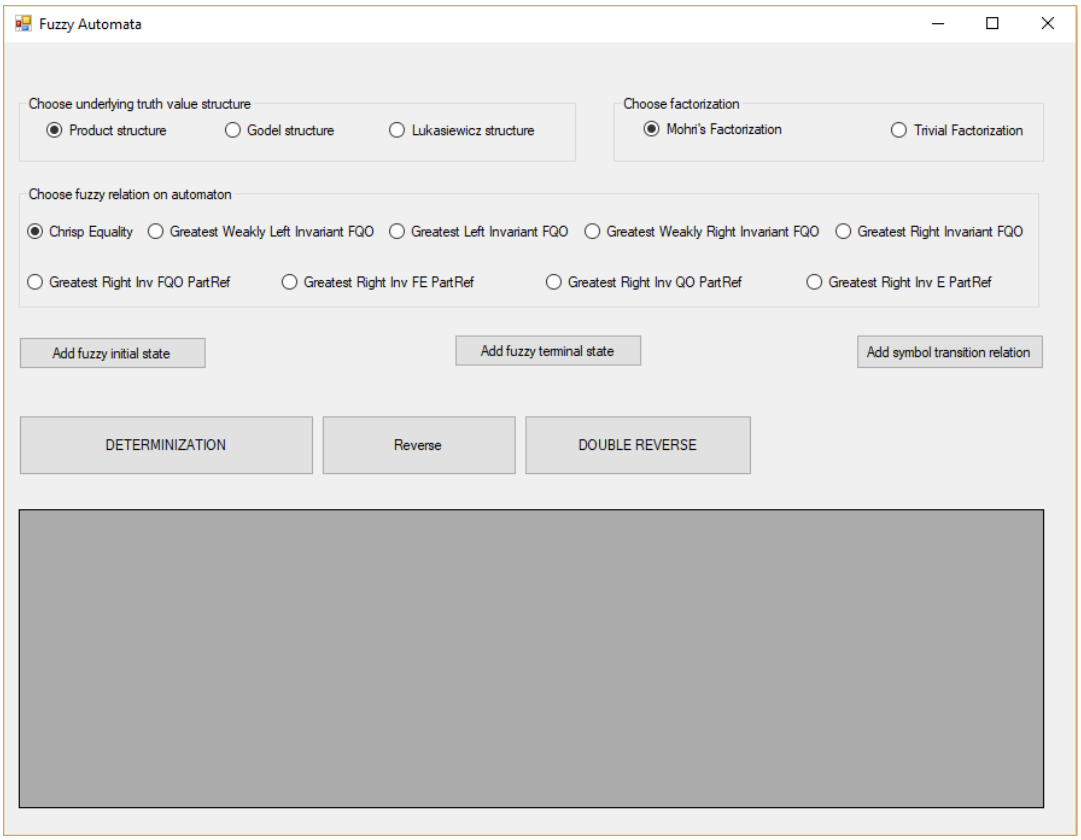
Implementacija algoritama za fazi automate

Dijagram klasa programa prikazan je na Slici A.2. Glavne komponente dijagrama klasa su:

- Apstraktna klasa CRLATTICE sa apstraktnim metodama MULT i RES za operacije množenja i reziduuma u kompletnoj reziduiranoj mreži, respektivno, zajedno sa tri izvedene klase CRLATTICELUKASIEWICZ, CRLATTICEPRODUCT i CRLATTICEGODEL, pri čemu svaka konkretna klasa implementira množenje i reziduum u standardnoj Łukasiewiczevoj, produkt i Gödelovoj algebri, respektivno.
- Apstraktna klasa FACTORIZATION sa apstraktnim metodama F i G za par funkcija $D = (f, g)$ koje formiraju faktorizaciju, zajedno sa dve izvedene klase FACTORIZATIONTRIVIAL i FACTORIZATIONMOHRI koje implementiraju funkcije f i g iz trivijalne i Mohrieve faktorizacije, respektivno.
- Klasa FUZZYSET koja sadrži tri atributa: NUMBEROFELEMENTS (ceo broj koji pamti broj elemenata fazi skupa), VALUES (vektor vrednosti fazi skupa), i TRUTHVALUE (tipa CRLATTICE). Takođe, preklapaju se sledeći operatori: $*$ (za kompoziciju dva fazi skupa), $\sim$ (za nalaženje supremuma među vrednostima fazi skupa), kao i logički operatori $==$ i $!=$.
- Klasa FUZZYRELATION koja sadrži tri atributa: NUMBEROFELEMENTS (ceo broj koji pamti broj elemenata fazi relacije), VALUES (matrica vrednosti fazi relacija), i TRUTHVALUE (tipa CRLATTICE). Takođe, preklapaju se sledeći operatori: $*$ (za kompoziciju fazi relacija, kao i kompoziciju fazi skupa i fazi relacije), kao i logički operatori $==$ i $!=$.
- Klasa FORM1 predstavlja jezgro programa. Od atributa sadrži INITALSTATE and TERMINALSTATES (tipa FUZZYSTATE), TRANSITIONRELATIONS (niz elemenata tipa FUZZYRELATIONS), i metode za determinizaciju prikazanih u disertaciji, kao i za računanje najvećih (slabo) desno i levo invarijatnih fazi ekvivalencija i fazi kvazi-uređenja.
- Na kraju, imamo pomoćne klase PROMPT za generisanje poruka korisniku, kao i klasu PROGRAM koja pokreće program, a koristi klasu *Form1*.

IZVORNI KÔD FAJLA A.1: Codes/FuzzyAutomata/CRLattice.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
```



SLIKA A.1: Grafički prikaz aplikacije.

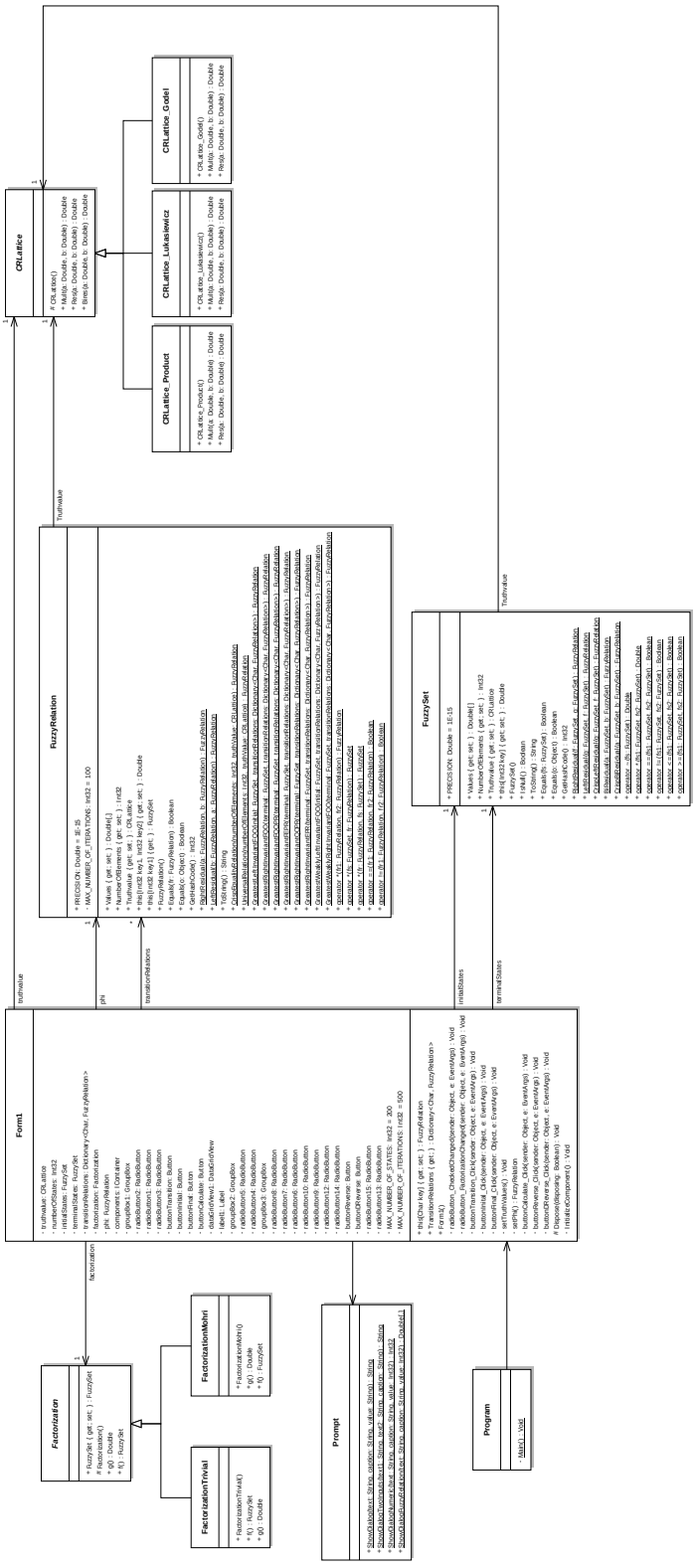
```
using System.Text;
using System.Threading.Tasks;

namespace FuzzyAutomata
{
    public abstract class CRLattice
    {
        public abstract double Mult(double a, double b); // Multiplication
        public abstract double Res(double a, double b); // Residuum
        public double Bires(double a, double b) // Biresiduum
        {
            return Math.Min(Res(a, b), Res(b, a));
        }
    }
}
```

IZVORNI KÔD FAJLA A.2: Codes/FuzzyAutomata/CRLattice_Godel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FuzzyAutomata
{
    class CRLattice_Godel : CRLattice
    {
        public override double Mult(double a, double b)
        {
            return Math.Min(a, b);
```



SLIKA A.2: Pregled dijagrama klasa.

```
14          }
15
16          public override double Res(double a, double b)
17          {
18              return (a <= b ? 1 : b);
19          }
20      }
21  }
```

IZVORNI KÔD FAJLA A.3: Codes/FuzzyAutomata/CRLattice_Product.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FuzzyAutomata
8  {
9      class CRLattice_Product : CRLattice
10     {
11         public override double Mult(double a, double b)
12         {
13             return a * b;
14         }
15
16         public override double Res(double a, double b)
17         {
18             return (a <= b ? 1 : b / a);
19         }
20     }
21 }
```

IZVORNI KÔD FAJLA A.4: Codes/FuzzyAutomata/CRLattice_Lukasiewicz.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FuzzyAutomata
8  {
9      class CRLattice_Lukasiewicz : CRLattice
10     {
11         public override double Mult(double a, double b)
12         {
13             return Math.Max(a + b - 1, 0);
14         }
15
16         public override double Res(double a, double b)
17         {
18             return Math.Min(1, 1 - a + b);
19         }
20     }
21 }
```

IZVORNI KÔD FAJLA A.5: Codes/FuzzyAutomata/Factorization.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
```

```
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FuzzyAutomata
8  {
9      abstract public class Factorization
10     {
11         public FuzzySet FuzzySet { get; set; }
12
13         public abstract double g();
14         public abstract FuzzySet f();
15     }
16 }
```

IZVORNI KÔD FAJLA A.6: Codes/FuzzyAutomata/FactorizationMohri.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FuzzyAutomata
8  {
9      public class FactorizationMohri : Factorization
10     {
11         public override double g()
12         {
13             return FuzzySet.IsNull() ? 1 : ~FuzzySet;
14         }
15
16         public override FuzzySet f()
17         {
18             int n = FuzzySet.NumberOfElements;
19             FuzzySet result = new FuzzySet() { NumberOfElements = n,
                   Truthvalue = FuzzySet.Truthvalue };
20             result.Values = new double[n];
21             for (int i = 0; i < n; i++)
22             {
23                 result[i] = FuzzySet.Truthvalue.Res(g(), FuzzySet[i]);
24             }
25             return result;
26         }
27     }
28 }
```

IZVORNI KÔD FAJLA A.7: Codes/FuzzyAutomata/FactorizationTrivial.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace FuzzyAutomata
8  {
9      class FactorizationTrivial : Factorization
10     {
11         public override FuzzySet f()
12         {
13             return FuzzySet;
14         }
15
```

```
16          public override double g()
17          {
18              return 1;
19          }
20      }
21  }
```

IZVORNI KÔD FAJLA A.8: Codes/FuzzyAutomata/FuzzySet.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace FuzzyAutomata
8   {
9       public class FuzzySet
10      {
11          public const double PRECISION = 1E-15;
12
13          public double[] Values { get; set; } // Values of fuzzy set
                (vector)
14          public int NumberOfElements { get; set; } // Total number of
                values
15          public CRLattice Truthvalue { get; set; } // Truth Value for
                fuzzy set
16
17          /* Get element of fuzzy set with specific key */
18          public double this[int key]
19          {
20              get
21              {
22                  return Values[key];
23              }
24              set
25              {
26                  Values[key] = value;
27              }
28          }
29
30          /* Check if fuzzy set is empty */
31          public bool IsNull()
32          {
33              for(int i = 0; i < NumberOfElements; i++)
34              {
35                  if (this[i] != 0) return false;
36              }
37              return true;
38          }
39
40          /* Write fuzzy set as string */
41          public override string ToString()
42          {
43              return "{" + string.Join(", ", Values) + "}";
44          }
45
46          /* Find supremum of fuzzy set */
47          public static double operator ~(FuzzySet fs)
48          {
49              double res = fs[0];
50              for (int i = 0; i < fs.NumberOfElements; i++)
51              {
```

```
52                  if (fs[i] > res) res = fs[i];
53              }
54              return res;
55          }
56
57          /* Composition of two fuzzy sets */
58          public static double operator *(FuzzySet fs1, FuzzySet fs2)
59          {
60              if (fs1.NumberOfElements != fs2.NumberOfElements) return 0;
61              if (fs1.Truthvalue.GetType() != fs2.Truthvalue.GetType())
                    return 0;
62              int n = fs1.NumberOfElements;
63              double res = double.MinValue;
64              for (int i = 0; i < n; i++)
65              {
66                  res = Math.Max(res, fs1.Truthvalue.Mult(fs1[i], fs2[i]));
67              }
68              return res;
69          }
70
71          /* Check if two fuzzy sets are equal */
72          public static bool operator ==(FuzzySet fs1, FuzzySet fs2)
73          {
74              //if ((object)fs1 == null) return (object)fs2 == null;
75              //if ((object)fs2 == null) return (object)fs1 == null;
76              return fs1.Equals(fs2);
77          }
78
79          /* Check if two fuzzy sets are not equal */
80          public static bool operator !=(FuzzySet fs1, FuzzySet fs2)
81          {
82              if ((object)fs1 == null) return (object)fs2 != null;
83              if ((object)fs2 == null) return (object)fs1 != null;
84              return !fs1.Equals(fs2);
85          }
86
87          /* Check the ordering of fuzzy sets */
88          public static bool operator <=(FuzzySet fs1, FuzzySet fs2)
89          {
90              if (fs1.Truthvalue != fs2.Truthvalue) return false;
91              if (fs1.NumberOfElements != fs2.NumberOfElements) return
                    false;
92              for(int i = 0; i < fs1.NumberOfElements; i++)
93              {
94                  if (fs1[i] > fs2[i]) return false;
95              }
96              return true;
97          }
98
99          public static bool operator >=(FuzzySet fs1, FuzzySet fs2)
100         {
101             if (fs1.Truthvalue != fs2.Truthvalue) return false;
102             if (fs1.NumberOfElements != fs2.NumberOfElements) return
                    false;
103             for (int i = 0; i < fs1.NumberOfElements; i++)
104             {
105                 if (fs1[i] < fs2[i]) return false;
106             }
107             return true;
108         }
109
110         public bool Equals(FuzzySet fs)
111         {
```

```
112             //if ((object)fs == null) return false;
113             if (Truthvalue.GetType() != fs.Truthvalue.GetType()) return
                    false;
114             if (NumberOfElements != fs.NumberOfElements) return false;
115             for (int i = 0; i < NumberOfElements; i++)
116             {
117                 if (Math.Abs(this[i] - fs[i]) > PRECISION) return false;
118             }
119             return true;
120         }
121 
122         public override bool Equals(object o)
123         {
124             return Equals(o as FuzzySet);
125         }
126 
127         public override int GetHashCode()
128         {
129             return NumberOfElements;
130         }
131 
132         /* Compute right residual of two fuzzy sets */
133         public static FuzzyRelation RightResidual(FuzzySet f, FuzzySet g)
134         {
135             int n = f.NumberOfElements;
136             double[,] res = new double[n, n];
137             for(int i = 0; i < n; i++)
138             {
139                 for(int j = 0; j < n; j++)
140                 {
141                     res[i, j] = f.Truthvalue.Res(f[i], g[j]);
142                 }
143             }
144             return new FuzzyRelation() { NumberOfElements = n, Truthvalue
                    = f.Truthvalue, Values = res };
145         }
146 
147         /* Compute left residual of two fuzzy sets */
148         public static FuzzyRelation LeftResidual(FuzzySet g, FuzzySet f)
149         {
150             int n = f.NumberOfElements;
151             double[,] res = new double[n, n];
152             for (int i = 0; i < n; i++)
153             {
154                 for (int j = 0; j < n; j++)
155                 {
156                     res[j, i] = f.Truthvalue.Res(f[i], g[j]);
157                 }
158             }
159             return new FuzzyRelation() { NumberOfElements = n, Truthvalue
                    = f.Truthvalue, Values = res };
160         }
161 
162         /* Compute crisp left residual of two fuzzy sets */
163         public static FuzzyRelation CrispLeftResidual(FuzzySet g,
                FuzzySet f)
164         {
165             int n = f.NumberOfElements;
166             FuzzyRelation res = new FuzzyRelation() { NumberOfElements =
                    n, Truthvalue = f.Truthvalue, Values = new double[n, n] };
167             for (int i = 0; i < n; i++)
168             {
169                 for (int j = 0; j < n; j++)
```

```
170                 {
171                     res[j, i] = (f[i] <= g[j]) ? 1 : 0;
172                 }
173             }
174             return res;
175         }
176 
177         /* Compute biresidual of two fuzzy sets */
178         public static FuzzyRelation BiResidual(FuzzySet a, FuzzySet b)
179         {
180             int n = a.NumberOfElements;
181             FuzzyRelation res = new FuzzyRelation() { NumberOfElements =
                    n, Truthvalue = a.Truthvalue, Values = new double[n, n] };
182             for (int i = 0; i < n; i++)
183             {
184                 for (int j = 0; j < n; j++)
185                 {
186                     res[i, j] = res.Truthvalue.Bires(a[i], b[j]);
187                 }
188             }
189             return res;
190         }
191 
192         /* Compute crisp biresidual of two fuzzy sets */
193         public static FuzzyRelation CrispBiResidual(FuzzySet a, FuzzySet
                b)
194         {
195             int n = a.NumberOfElements;
196             FuzzyRelation res = new FuzzyRelation() { NumberOfElements =
                    n, Truthvalue = a.Truthvalue, Values = new double[n, n] };
197             for (int i = 0; i < n; i++)
198             {
199                 for (int j = 0; j < n; j++)
200                 {
201                     res[i, j] = (a[i] == b[j]) ? 1 : 0;
202                 }
203             }
204             return res;
205         }
206     }
207 }
```

IZVORNI KÔD FAJLA A.9: Codes/FuzzyAutomata/FuzzyRelation.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7 
8  namespace FuzzyAutomata
9  {
10     public class FuzzyRelation
11     {
12         public const double PRECISION = 1E-15;
13 
14         public double[,] Values { get; set; } // Values of fuzzy relation
               (matrix)
15         public int NumberOfElements { get; set; } // Total number of
               elements
16         public CRLattice Truthvalue { get; set; } // Truth Value for
               fuzzy relation
```

```
17
18          /* Get value at a specific position */
19          public double this[int key1, int key2]
20          {
21              get
22              {
23                  return Values[key1, key2];
24              }
25              set
26              {
27                  Values[key1, key2] = value;
28              }
29          }
30
31          /* Get a foreset of a fuzzy relation */
32          public FuzzySet this[int key1]
33          {
34              get
35              {
36                  double[] res = new double[NumberOfElements];
37                  for (int i = 0; i < NumberOfElements; i++)
38                  {
39                          res[i] = Values[i, key1];
40                  }
41                  return new FuzzySet() { NumberOfElements =
                        NumberOfElements, Truthvalue = Truthvalue, Values =
                        res };
42              }
43          }
44
45          /* Compute composition of fuzzy relations */
46          public static FuzzyRelation operator *(FuzzyRelation fr1,
                FuzzyRelation fr2)
47          {
48              if (fr1.NumberOfElements != fr2.NumberOfElements) return null;
49              if (fr1.Truthvalue.GetType() != fr2.Truthvalue.GetType())
                    return null;
50              int n = fr1.NumberOfElements;
51              double[,] res = new double[n, n];
52              for (int i = 0; i < n; i++)
53              {
54                  for (int j = 0; j < n; j++)
55                  {
56                      double max = double.MinValue;
57                      for (int k = 0; k < n; k++)
58                      {
59                          max = Math.Max(max, fr1.Truthvalue.Mult(fr1[i,
                                k], fr2[k, j]));
60                      }
61                      res[i, j] = max;
62                  }
63              }
64              return new FuzzyRelation() { NumberOfElements = n, Truthvalue
                    = fr1.Truthvalue, Values = res };
65          }
66
67          /* Compute composition of fuzzy set and fuzzy relation */
68          public static FuzzySet operator *(FuzzySet fs, FuzzyRelation fr)
69          {
70              if (fs.NumberOfElements != fr.NumberOfElements) return null;
71              if (fs.Truthvalue.GetType() != fr.Truthvalue.GetType())
                    return null;
72              int n = fs.NumberOfElements;
```

```
73              double[] res = new double[n];
74              for (int j = 0; j < n; j++)
75              {
76                  double max = double.MinValue;
77                  for (int k = 0; k < n; k++)
78                  {
79                      max = Math.Max(max, fs.Truthvalue.Mult(fs[k], fr[k,
                            j]));
80                  }
81                  res[j] = max;
82              }
83              return new FuzzySet() { NumberOfElements = n, Truthvalue =
                    fs.Truthvalue, Values = res };
84          }
85
86          /* Compute composition of fuzzy relation and fuzzy set */
87          public static FuzzySet operator *(FuzzyRelation fr, FuzzySet fs)
88          {
89              if (fr.NumberOfElements != fs.NumberOfElements) return null;
90              if (fr.Truthvalue.GetType() != fs.Truthvalue.GetType())
                    return null;
91              int n = fr.NumberOfElements;
92              double[] res = new double[n];
93              for (int i = 0; i < n; i++)
94              {
95                  double max = double.MinValue;
96                  for (int k = 0; k < n; k++)
97                  {
98                      max = Math.Max(max, fr.Truthvalue.Mult(fr[i, k],
                            fs[k]));
99                  }
100                 res[i] = max;
101             }
102             return new FuzzySet() { NumberOfElements = n, Truthvalue =
                    fr.Truthvalue, Values = res };
103         }
104
105         /* Check if two fuzzy relations are equal */
106         public static bool operator ==(FuzzyRelation fr1, FuzzyRelation
                fr2)
107         {
108             return fr1.Equals(fr2);
109         }
110
111         /* Check if two fuzzy relations are not equal */
112         public static bool operator !=(FuzzyRelation fr1, FuzzyRelation
                fr2)
113         {
114             if ((object)fr1 == null) return (object)fr2 != null;
115             if ((object)fr2 == null) return (object)fr1 != null;
116             return !fr1.Equals(fr2);
117         }
118
119         public bool Equals(FuzzyRelation fr)
120         {
121             if (Truthvalue.GetType() != fr.Truthvalue.GetType()) return
                    false;
122             if (NumberOfElements != fr.NumberOfElements) return false;
123             for (int i = 0; i < NumberOfElements; i++)
124             {
125                 for (int j = 0; j < NumberOfElements; j++)
126                 {
127                     //if (this[i, j] != fr[i, j]) return false;
```

```
128                     if (Math.Abs(this[i, j] - fr[i, j]) > PRECISION)
                            return false;
129                 }
130             }
131             return true;
132         }
133 
134         public override bool Equals(object o)
135         {
136             return Equals(o as FuzzyRelation);
137         }
138 
139         public override int GetHashCode()
140         {
141             return NumberOfElements;
142         }
143 
144         /* Compute right residual of two fuzzy relations */
145         public static FuzzyRelation RightResidual(FuzzyRelation a,
                FuzzyRelation b)
146         {
147             int n = a.NumberOfElements;
148             FuzzyRelation res = new FuzzyRelation() { NumberOfElements =
                    n, Truthvalue = a.Truthvalue, Values = new double[n, n] };
149             for (int i = 0; i < n; i++)
150             {
151                 for (int j = 0; j < n; j++)
152                 {
153                     res[i, j] = double.MaxValue;
154                     for (int k = 0; k < n; k++)
155                     {
156                         res[i, j] = Math.Min(res[i, j],
                                a.Truthvalue.Res(a[k, i], b[k, j]));
157                     }
158                 }
159             }
160             return res;
161         }
162 
163         /* Compute left residual of two fuzzy relations */
164         public static FuzzyRelation LeftResidual(FuzzyRelation b,
                FuzzyRelation a)
165         {
166             int n = a.NumberOfElements;
167             FuzzyRelation res = new FuzzyRelation() { NumberOfElements =
                    n, Truthvalue = a.Truthvalue, Values = new double[n, n] };
168             for (int i = 0; i < n; i++)
169             {
170                 for (int j = 0; j < n; j++)
171                 {
172                     res[i, j] = double.MaxValue;
173                     for (int k = 0; k < n; k++)
174                     {
175                         res[i, j] = Math.Min(res[i, j],
                                a.Truthvalue.Res(a[j, k], b[i, k]));
176                     }
177                 }
178             }
179             return res;
180         }
181 
182         /* Convert fuzzy relation to string */
183         public override string ToString()
```

```
184         {
185             string result = "";
186             for (int i = 0; i < NumberOfElements; i++)
187             {
188                 for (int j = 0; j < NumberOfElements; j++)
189                 {
190                     result += this[i, j] + " ";
191                 }
192                 result += "\n";
193             }
194             result += "\n";
195             return result;
196         }
197 
198         /* Construct crisp equality relation */
199         public static FuzzyRelation CrispEqualityRelation(int
                numberOfElements, CRLattice truthValue)
200         {
201             double[,] res = new double[numberOfElements,
                    numberOfElements];
202             for (int i = 0; i < numberOfElements; i++)
203             {
204                 for (int j = 0; j < numberOfElements; j++)
205                 {
206                     res[i, j] = (i == j ? 1 : 0);
207                 }
208             }
209             return new FuzzyRelation() { NumberOfElements =
                    numberOfElements, Truthvalue = truthValue, Values = res };
210         }
211 
212         /* Construct universal relation */
213         public static FuzzyRelation UniversalRelation(int
                numberOfElements, CRLattice truthValue)
214         {
215             double[,] res = new double[numberOfElements,
                    numberOfElements];
216             for (int i = 0; i < numberOfElements; i++)
217             {
218                 for (int j = 0; j < numberOfElements; j++)
219                 {
220                     res[i, j] = 1;
221                 }
222             }
223             return new FuzzyRelation() { NumberOfElements =
                    numberOfElements, Truthvalue = truthValue, Values = res };
224         }
225 
226         const int MAX_NUMBER_OF_ITERATIONS = 100;
227 
228         /* Compute greatest left invariant fuzzy quasi-order (by
                approximating fixpoint) */
229         public static FuzzyRelation GreatestLeftInvariantFQO(FuzzySet
                initial, Dictionary<char, FuzzyRelation> transitionRelations)
230         {
231             int n = initial.NumberOfElements;
232             StreamWriter sw = new StreamWriter("debug.txt");
233             FuzzyRelation phi = FuzzySet.RightResidual(initial, initial);
234             FuzzyRelation oldPhi, newPhi;
235             int numberOfIterations = 0;
236             do
237             {
238                 /*** Write Phi in debug file ****/
```

```
239                 for (int i = 0; i < n; i++)
240                 {
241                     for (int j = 0; j < n; j++)
242                     {
243                         sw.Write(phi[i, j] + " ");
244                     }
245                     sw.WriteLine();
246                 }
247                 sw.WriteLine();
248                 /*** End wirte Phi ****/
249 
250                 double[,] calc = new double[n, n];
251                 for(int i = 0; i < n; i++)
252                 {
253                     for(int j = 0; j < n; j++)
254                     {
255                         calc[i, j] = phi[i, j];
256                     }
257                 }
258                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
259                 {
260                     FuzzyRelation tmp = FuzzyRelation.RightResidual(phi *
                            transitionRelation.Value, phi *
                            transitionRelation.Value);
261                     for (int i = 0; i < n; i++)
262                     {
263                         for (int j = 0; j < n; j++)
264                         {
265                             calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
266                         }
267                     }
268                 }
269                 newPhi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
270                 oldPhi = phi;
271                 phi = newPhi;
272             }
273             while ((oldPhi != newPhi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS));
274             sw.Close();
275             return newPhi;
276         }
277 
278         /* Compute greatest right invariant fuzzy quasi-order (by
                approximating fixpoint) */
279         public static FuzzyRelation GreatestRightInvariantFQO(FuzzySet
                terminal, Dictionary<char, FuzzyRelation> transitionRelations)
280         {
281             int n = terminal.NumberOfElements;
282             StreamWriter sw = new StreamWriter("debug.txt");
283             FuzzyRelation phi = FuzzySet.LeftResidual(terminal, terminal);
284             FuzzyRelation oldPhi, newPhi;
285             int numberOfIterations = 0;
286             do
287             {
288                 /*** Write Phi in debug file ****/
289                 for (int i = 0; i < n; i++)
290                 {
291                     for (int j = 0; j < n; j++)
292                     {
293                         sw.Write(phi[i, j] + " ");
294                     }
```

```
295                     sw.WriteLine();
296                 }
297                 sw.WriteLine();
298                 /*** End wirte Phi ****/
299 
300                 double[,] calc = new double[n, n];
301                 for (int i = 0; i < n; i++)
302                 {
303                     for (int j = 0; j < n; j++)
304                     {
305                         calc[i, j] = phi[i, j];
306                     }
307                 }
308                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
309                 {
310                     FuzzyRelation tmp =
                            FuzzyRelation.LeftResidual(transitionRelation.Value
                            * phi, transitionRelation.Value * phi);
311                     for (int i = 0; i < n; i++)
312                     {
313                         for (int j = 0; j < n; j++)
314                         {
315                             calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
316                         }
317                     }
318                 }
319                 newPhi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
320                 oldPhi = phi;
321                 phi = newPhi;
322             }
323             while ((oldPhi != newPhi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS));
324             sw.Close();
325             return newPhi;
326         }
327 
328         /* Compute greatest right invariant fuzzy quasi-order (by
                partition refinement) */
329         public static FuzzyRelation GreatestRightInvariantFQOPR(FuzzySet
                terminal, Dictionary<char, FuzzyRelation> transitionRelations)
330         {
331             int n = terminal.NumberOfElements;
332             StreamWriter sw = new StreamWriter("debug.txt");
333             FuzzyRelation phi = FuzzyRelation.UniversalRelation(n,
                    terminal.Truthvalue);
334             FuzzyRelation varphi = FuzzySet.LeftResidual(terminal,
                    terminal);
335             double[,] calc = new double[n, n];
336             for (int i = 0; i < n; i++)
337             {
338                 for (int j = 0; j < n; j++)
339                 {
340                     calc[i, j] = varphi[i, j];
341                 }
342             }
343             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
344             {
345                 FuzzyRelation tmp =
                        FuzzySet.LeftResidual(transitionRelation.Value *
                        phi[0], transitionRelation.Value * phi[0]);
```

```
346                 for (int i = 0; i < n; i++)
347                 {
348                     for (int j = 0; j < n; j++)
349                     {
350                         calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
351                     }
352                 }
353             }
354             varphi = new FuzzyRelation() { NumberOfElements = n,
                    Truthvalue = phi.Truthvalue, Values = calc };
355
356             /*** Write Phi in debug file ****/
357             sw.Write("Phi[0] = ");
358             for (int i = 0; i < n; i++)
359             {
360                 for (int j = 0; j < n; j++)
361                 {
362                     sw.Write(phi[i, j] + " ");
363                 }
364                 sw.WriteLine();
365             }
366             sw.WriteLine();
367             /*** End write Phi ****/
368             /*** Write Varphi in debug file ****/
369             sw.Write("Varphi[0] = ");
370             for (int i = 0; i < n; i++)
371             {
372                 for (int j = 0; j < n; j++)
373                 {
374                     sw.Write(varphi[i, j] + " ");
375                 }
376                 sw.WriteLine();
377             }
378             sw.WriteLine();
379             /*** End write Varphi ****/
380
381             int numberOfIterations = 0;
382             while ((phi != varphi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS))
383             {
384                 int index = 0;
385                 while ((phi[index] == varphi[index]) && (index < n))
386                 {
387                     index++;
388                 }
389                 FuzzyRelation tmp = FuzzySet.LeftResidual(varphi[index],
                        varphi[index]);
390                 for (int i = 0; i < n; i++)
391                 {
392                     for (int j = 0; j < n; j++)
393                     {
394                         phi[i, j] = Math.Min(phi[i, j], tmp[i, j]);
395                     }
396                 }
397                 calc = new double[n, n];
398                 for (int i = 0; i < n; i++)
399                 {
400                     for (int j = 0; j < n; j++)
401                     {
402                         calc[i, j] = varphi[i, j];
403                     }
404                 }
```

```
405                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
406                 {
407                     for(int m = 0; m < n; m++)
408                     {
409                         if(varphi[m, index] != 0)
410                         {
411                             tmp =
                                FuzzySet.LeftResidual(transitionRelation.Value
                                * phi[m], transitionRelation.Value *
                                phi[m]);
412                             for (int i = 0; i < n; i++)
413                             {
414                                 for (int j = 0; j < n; j++)
415                                 {
416                                     calc[i, j] = Math.Min(calc[i, j],
                                        tmp[i, j]);
417                                 }
418                             }
419                         }
420                     }
421                 }
422                 varphi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
423                 /*** Write Phi in debug file ****/
424                 sw.Write("Phi[{0}] = ", numberOfIterations);
425                 for (int i = 0; i < n; i++)
426                 {
427                     for (int j = 0; j < n; j++)
428                     {
429                         sw.Write(phi[i, j] + " ");
430                     }
431                     sw.WriteLine();
432                 }
433                 sw.WriteLine();
434                 /*** End write Phi ****/
435                 /*** Write Varphi in debug file ****/
436                 sw.Write("Varphi[{0}] = ", numberOfIterations);
437                 for (int i = 0; i < n; i++)
438                 {
439                     for (int j = 0; j < n; j++)
440                     {
441                         sw.Write(varphi[i, j] + " ");
442                     }
443                     sw.WriteLine();
444                 }
445                 sw.WriteLine();
446                 /*** End write Varphi ****/
447             }
448             sw.Close();
449             return varphi;
450         }
451
452         /* Compute greatest right invariant fuzzy equivalence (by
                partition refinement) */
453         public static FuzzyRelation GreatestRightInvariantFEPR(FuzzySet
                terminal, Dictionary<char, FuzzyRelation> transitionRelations)
454         {
455             int n = terminal.NumberOfElements;
456             StreamWriter sw = new StreamWriter("debug.txt");
457             FuzzyRelation phi = FuzzyRelation.UniversalRelation(n,
                    terminal.Truthvalue);
```

```
458             FuzzyRelation varphi = FuzzySet.BiResidual(terminal,
                    terminal);
459             double[,] calc = new double[n, n];
460             for (int i = 0; i < n; i++)
461             {
462                 for (int j = 0; j < n; j++)
463                 {
464                     calc[i, j] = varphi[i, j];
465                 }
466             }
467             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
468             {
469                 FuzzyRelation tmp =
                        FuzzySet.BiResidual(transitionRelation.Value * phi[0],
                        transitionRelation.Value * phi[0]);
470                 for (int i = 0; i < n; i++)
471                 {
472                     for (int j = 0; j < n; j++)
473                     {
474                         calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
475                     }
476                 }
477             }
478             varphi = new FuzzyRelation() { NumberOfElements = n,
                    Truthvalue = phi.Truthvalue, Values = calc };
479
480             /*** Write Phi in debug file ****/
481             sw.Write("Phi[0] = ");
482             for (int i = 0; i < n; i++)
483             {
484                 for (int j = 0; j < n; j++)
485                 {
486                     sw.Write(phi[i, j] + " ");
487                 }
488                 sw.WriteLine();
489             }
490             sw.WriteLine();
491             /*** End write Phi ****/
492             /*** Write Varphi in debug file ****/
493             sw.Write("Varphi[0] = ");
494             for (int i = 0; i < n; i++)
495             {
496                 for (int j = 0; j < n; j++)
497                 {
498                     sw.Write(varphi[i, j] + " ");
499                 }
500                 sw.WriteLine();
501             }
502             sw.WriteLine();
503             /*** End write Varphi ****/
504
505             int numberOfIterations = 0;
506             while ((phi != varphi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS))
507             {
508                 int index = 0;
509                 while ((phi[index] == varphi[index]) && (index++ < n)) ;
510                 FuzzyRelation tmp = FuzzySet.BiResidual(varphi[index],
                        varphi[index]);
511                 for (int i = 0; i < n; i++)
512                 {
513                     for (int j = 0; j < n; j++)
```

```
514                     {
515                         phi[i, j] = Math.Min(phi[i, j], tmp[i, j]);
516                     }
517                 }
518                 calc = new double[n, n];
519                 for (int i = 0; i < n; i++)
520                 {
521                     for (int j = 0; j < n; j++)
522                     {
523                         calc[i, j] = varphi[i, j];
524                     }
525                 }
526 
527                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
528                 {
529                     tmp = FuzzySet.BiResidual(transitionRelation.Value *
                            varphi[index], transitionRelation.Value *
                            varphi[index]);
530                     for (int i = 0; i < n; i++)
531                     {
532                         for (int j = 0; j < n; j++)
533                         {
534                             calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
535                         }
536                     }
537                 }
538 
539                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
540                 {
541                     for (int m = 0; m < n; m++)
542                     {
543                         if (varphi[m, index] != 1)
544                         {
545                             tmp =
                                    FuzzySet.BiResidual(transitionRelation.Value
                                    * phi[m], transitionRelation.Value *
                                    phi[m]);
546                             for (int i = 0; i < n; i++)
547                             {
548                                 for (int j = 0; j < n; j++)
549                                 {
550                                     calc[i, j] = Math.Min(calc[i, j],
                                            tmp[i, j]);
551                                 }
552                             }
553                         }
554                     }
555                 }
556                 varphi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
557                 /*** Write Phi in debug file ****/
558                 sw.Write("Phi[{0}] = ", numberOfIterations);
559                 for (int i = 0; i < n; i++)
560                 {
561                     for (int j = 0; j < n; j++)
562                     {
563                         sw.Write(phi[i, j] + " ");
564                     }
565                     sw.WriteLine();
566                 }
567                 sw.WriteLine();
```

```
568                 /*** End write Phi ****/
569                 /*** Write Varphi in debug file ****/
570                 sw.Write("Varphi[{0}] = ", numberOfIterations);
571                 for (int i = 0; i < n; i++)
572                 {
573                     for (int j = 0; j < n; j++)
574                     {
575                         sw.Write(varphi[i, j] + " ");
576                     }
577                     sw.WriteLine();
578                 }
579                 sw.WriteLine();
580                 /*** End write Varphi ****/
581             }
582             sw.Close();
583             return varphi;
584         }
585 
586         /* Compute greatest right invariant crisp quasi-order (by
                partition refinement) */
587         public static FuzzyRelation GreatestRightInvariantQOPR(FuzzySet
                terminal, Dictionary<char, FuzzyRelation> transitionRelations)
588         {
589             int n = terminal.NumberOfElements;
590             StreamWriter sw = new StreamWriter("debug.txt");
591             FuzzyRelation phi = FuzzyRelation.UniversalRelation(n,
                    terminal.Truthvalue);
592             FuzzyRelation varphi = FuzzySet.CrispLeftResidual(terminal,
                    terminal);
593             double[,] calc = new double[n, n];
594             for (int i = 0; i < n; i++)
595             {
596                 for (int j = 0; j < n; j++)
597                 {
598                     calc[i, j] = varphi[i, j];
599                 }
600             }
601             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
602             {
603                 FuzzyRelation tmp =
                        FuzzySet.CrispLeftResidual(transitionRelation.Value *
                        phi[0], transitionRelation.Value * phi[0]);
604                 for (int i = 0; i < n; i++)
605                 {
606                     for (int j = 0; j < n; j++)
607                     {
608                         calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
609                     }
610                 }
611             }
612             varphi = new FuzzyRelation() { NumberOfElements = n,
                    Truthvalue = phi.Truthvalue, Values = calc };
613 
614             /*** Write Phi in debug file ****/
615             sw.Write("Phi[0] = ");
616             for (int i = 0; i < n; i++)
617             {
618                 for (int j = 0; j < n; j++)
619                 {
620                     sw.Write(phi[i, j] + " ");
621                 }
622                 sw.WriteLine();
```

```
623             }
624             sw.WriteLine();
625             /*** End write Phi ****/
626             /*** Write Varphi in debug file ****/
627             sw.Write("Varphi[0] = ");
628             for (int i = 0; i < n; i++)
629             {
630                 for (int j = 0; j < n; j++)
631                 {
632                     sw.Write(varphi[i, j] + " ");
633                 }
634                 sw.WriteLine();
635             }
636             sw.WriteLine();
637             /*** End write Varphi ****/
638 
639             int numberOfIterations = 0;
640             while ((phi != varphi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS))
641             {
642                 int index = 0;
643                 while ((phi[index] == varphi[index]) && (index < n))
644                 {
645                     index++;
646                 }
647                 FuzzyRelation tmp =
                        FuzzySet.CrispLeftResidual(varphi[index],
                        varphi[index]);
648                 for (int i = 0; i < n; i++)
649                 {
650                     for (int j = 0; j < n; j++)
651                     {
652                         phi[i, j] = Math.Min(phi[i, j], tmp[i, j]);
653                     }
654                 }
655                 calc = new double[n, n];
656                 for (int i = 0; i < n; i++)
657                 {
658                     for (int j = 0; j < n; j++)
659                     {
660                         calc[i, j] = varphi[i, j];
661                     }
662                 }
663                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
664                 {
665                     for (int m = 0; m < n; m++)
666                     {
667                         if (varphi[m, index] != 0)
668                         {
669                             tmp =
                                    FuzzySet.CrispLeftResidual(transitionRelation.Value
                                    * phi[m], transitionRelation.Value *
                                    phi[m]);
670                             for (int i = 0; i < n; i++)
671                             {
672                                 for (int j = 0; j < n; j++)
673                                 {
674                                     calc[i, j] = Math.Min(calc[i, j],
                                            tmp[i, j]);
675                                 }
676                             }
677                         }
```

```
678                     }
679                 }
680                 varphi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
681                 /*** Write Phi in debug file ****/
682                 sw.Write("Phi[{0}] = ", numberOfIterations);
683                 for (int i = 0; i < n; i++)
684                 {
685                     for (int j = 0; j < n; j++)
686                     {
687                         sw.Write(phi[i, j] + " ");
688                     }
689                     sw.WriteLine();
690                 }
691                 sw.WriteLine();
692                 /*** End write Phi ****/
693                 /*** Write Varphi in debug file ****/
694                 sw.Write("Varphi[{0}] = ", numberOfIterations);
695                 for (int i = 0; i < n; i++)
696                 {
697                     for (int j = 0; j < n; j++)
698                     {
699                         sw.Write(varphi[i, j] + " ");
700                     }
701                     sw.WriteLine();
702                 }
703                 sw.WriteLine();
704                 /*** End write Varphi ****/
705             }
706             sw.Close();
707             return varphi;
708         }
709
710         /* Compute greatest right invariant crisp equivalence (by
                partition refinement) */
711         public static FuzzyRelation GreatestRightInvariantEPR(FuzzySet
                terminal, Dictionary<char, FuzzyRelation> transitionRelations)
712         {
713             int n = terminal.NumberOfElements;
714             StreamWriter sw = new StreamWriter("debug.txt");
715             FuzzyRelation phi = FuzzyRelation.UniversalRelation(n,
                    terminal.Truthvalue);
716             FuzzyRelation varphi = FuzzySet.CrispBiResidual(terminal,
                    terminal);
717             double[,] calc = new double[n, n];
718             for (int i = 0; i < n; i++)
719             {
720                 for (int j = 0; j < n; j++)
721                 {
722                     calc[i, j] = varphi[i, j];
723                 }
724             }
725             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
726             {
727                 FuzzyRelation tmp =
                        FuzzySet.CrispBiResidual(transitionRelation.Value *
                        phi[0], transitionRelation.Value * phi[0]);
728                 for (int i = 0; i < n; i++)
729                 {
730                     for (int j = 0; j < n; j++)
731                     {
732                         calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
```

```
733                     }
734                 }
735             }
736             varphi = new FuzzyRelation() { NumberOfElements = n,
                    Truthvalue = phi.Truthvalue, Values = calc };
737
738             /*** Write Phi in debug file ****/
739             sw.Write("Phi[0] = ");
740             for (int i = 0; i < n; i++)
741             {
742                 for (int j = 0; j < n; j++)
743                 {
744                     sw.Write(phi[i, j] + " ");
745                 }
746                 sw.WriteLine();
747             }
748             sw.WriteLine();
749             /*** End write Phi ****/
750             /*** Write Varphi in debug file ****/
751             sw.Write("Varphi[0] = ");
752             for (int i = 0; i < n; i++)
753             {
754                 for (int j = 0; j < n; j++)
755                 {
756                     sw.Write(varphi[i, j] + " ");
757                 }
758                 sw.WriteLine();
759             }
760             sw.WriteLine();
761             /*** End write Varphi ****/
762
763             int numberOfIterations = 0;
764             while ((phi != varphi) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS))
765             {
766                 int index = 0;
767                 while ((phi[index] == varphi[index]) && (index++ < n)) ;
768                 FuzzyRelation tmp =
                        FuzzySet.CrispBiResidual(varphi[index], varphi[index]);
769                 for (int i = 0; i < n; i++)
770                 {
771                     for (int j = 0; j < n; j++)
772                     {
773                         phi[i, j] = Math.Min(phi[i, j], tmp[i, j]);
774                     }
775                 }
776                 calc = new double[n, n];
777                 for (int i = 0; i < n; i++)
778                 {
779                     for (int j = 0; j < n; j++)
780                     {
781                         calc[i, j] = varphi[i, j];
782                     }
783                 }
784
785                 FuzzySet vect = new FuzzySet { NumberOfElements = n,
                        Truthvalue = terminal.Truthvalue, Values = new
                        double[n] };
786                 for (int i = 0; i < n; i++)
787                 {
788                     vect[i] = phi[index][i];
789                 }
```

```
790                 FuzzySet vect1 = new FuzzySet { NumberOfElements = n,
                        Truthvalue = terminal.Truthvalue, Values = new
                        double[n] };
791                 for (int i = 0; i < n; i++)
792                 {
793                     vect1[i] = phi[index][i] == 1 ? 0 : 1;
794                 }
795                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
796                 {
797
798                     tmp =
                            FuzzySet.CrispBiResidual(transitionRelation.Value
                            * vect, transitionRelation.Value * vect);
799                     for (int i = 0; i < n; i++)
800                     {
801                         for (int j = 0; j < n; j++)
802                         {
803                             calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
804                         }
805                     }
806                     tmp =
                            FuzzySet.CrispBiResidual(transitionRelation.Value
                            * vect1, transitionRelation.Value * vect1);
807                     for (int i = 0; i < n; i++)
808                     {
809                         for (int j = 0; j < n; j++)
810                         {
811                             calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
812                         }
813                     }
814                 }
815                 varphi = new FuzzyRelation() { NumberOfElements = n,
                        Truthvalue = phi.Truthvalue, Values = calc };
816                 /*** Write Phi in debug file ****/
817                 sw.Write("Phi[{0}] = ", numberOfIterations);
818                 for (int i = 0; i < n; i++)
819                 {
820                     for (int j = 0; j < n; j++)
821                     {
822                         sw.Write(phi[i, j] + " ");
823                     }
824                     sw.WriteLine();
825                 }
826                 sw.WriteLine();
827                 /*** End write Phi ****/
828                 /*** Write Varphi in debug file ****/
829                 sw.Write("Varphi[{0}] = ", numberOfIterations);
830                 for (int i = 0; i < n; i++)
831                 {
832                     for (int j = 0; j < n; j++)
833                     {
834                         sw.Write(varphi[i, j] + " ");
835                     }
836                     sw.WriteLine();
837                 }
838                 sw.WriteLine();
839                 /*** End write Varphi ****/
840             }
841             sw.Close();
842             return varphi;
843         }
844
```

```
845         /* Compute greatest weakly left invariant fuzzy quasi-order */
846         public static FuzzyRelation
                GreatestWeaklyLeftInvariantFQO(FuzzySet initial,
                Dictionary<char, FuzzyRelation> transitionRelations)
847         {
848             int n = initial.NumberOfElements;
849             StreamWriter sw = new StreamWriter("debug.txt");
850             Dictionary<string, FuzzySet> sigmas = new Dictionary<string,
                    FuzzySet>();
851             Queue<string> words = new Queue<string>();
852 
853             words.Enqueue("");
854             sigmas.Add("", initial);
855 
856             int numberOfIterations = 0;
857 
858             do
859             {
860                 string currentWord = words.Dequeue();
861                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
862                 {
863                     string newWord = currentWord + transitionRelation.Key;
864                     FuzzySet newSigma = sigmas[currentWord] *
                            transitionRelation.Value;
865                     bool alreadyCalc = false;
866                     foreach (KeyValuePair<string, FuzzySet> sigma in
                            sigmas)
867                     {
868                         if(sigma.Value == newSigma)
869                         {
870                             alreadyCalc = true;
871                             break;
872                         }
873                     }
874                     if(!alreadyCalc)
875                     {
876                         words.Enqueue(newWord);
877                         sigmas.Add(newWord, newSigma);
878                     }
879                 }
880             }
881             while ((words.Count > 0) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS));
882 
883             double[,] calc = new double[n, n];
884             for(int i = 0; i < n; i++)
885             {
886                 for(int j = 0; j < n; j++)
887                 {
888                     calc[i, j] = 1;
889                 }
890             }
891             foreach (KeyValuePair<string, FuzzySet> sigma in sigmas)
892             {
893                 /*** Write sigma in debug file ****/
894                 sw.Write("Sigma_{" + sigma.Key + "} = [");
895                 for(int i = 0; i < n; i++)
896                 {
897                     sw.Write(sigma.Value[i] + " ");
898                 }
899                 sw.WriteLine("]");
900                 /*** End wirte sigma ***/
```

```
901
902                 FuzzyRelation tmp = FuzzySet.RightResidual(sigma.Value,
                        sigma.Value);
903                 /*** Write tmp in debug file ****/
904                 for (int i = 0; i < n; i++)
905                 {
906                     for (int j = 0; j < n; j++)
907                     {
908                         sw.Write(tmp[i, j] + " ");
909                     }
910                     sw.WriteLine();
911                 }
912                 sw.WriteLine();
913                 /*** End wirte tmp ****/
914
915                 for (int i = 0; i < n; i++)
916                 {
917                     for (int j = 0; j < n; j++)
918                     {
919                         calc[i, j] = Math.Min(calc[i, j], tmp[i, j]);
920                     }
921                 }
922             }
923             sw.WriteLine();
924             /*** Write Phi in debug file ****/
925             for (int i = 0; i < n; i++)
926             {
927                 for (int j = 0; j < n; j++)
928                 {
929                     sw.Write(calc[i, j] + " ");
930                 }
931                 sw.WriteLine();
932             }
933             sw.WriteLine();
934             /*** End wirte Phi ****/
935
936             sw.Close();
937             return new FuzzyRelation() { NumberOfElements = n, Truthvalue
                    = initial.Truthvalue, Values = calc };
938         }
939
940         /* Compute greatest weakly right invariant fuzzy quasi-order */
941         public static FuzzyRelation
                GreatestWeaklyRightInvariantFQO(FuzzySet terminal,
                Dictionary<char, FuzzyRelation> transitionRelations)
942         {
943             int n = terminal.NumberOfElements;
944             StreamWriter sw = new StreamWriter("debug.txt");
945             Dictionary<string, FuzzySet> taus = new Dictionary<string,
                    FuzzySet>();
946             Queue<string> words = new Queue<string>();
947
948             words.Enqueue("");
949             taus.Add("", terminal);
950
951             int numberOfIterations = 0;
952
953             do
954             {
955                 string currentWord = words.Dequeue();
956                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
957                 {
```

```
958                     string newWord = currentWord + transitionRelation.Key;
959                     FuzzySet newTau = transitionRelation.Value *
                            taus[currentWord];
960                     bool alreadyCalc = false;
961                     foreach (KeyValuePair<string, FuzzySet> tau in taus)
962                     {
963                         if (tau.Value == newTau)
964                         {
965                             alreadyCalc = true;
966                             break;
967                         }
968                     }
969                     if (!alreadyCalc)
970                     {
971                         words.Enqueue(newWord);
972                         taus.Add(newWord, newTau);
973                     }
974                 }
975             }
976             while ((words.Count > 0) && (++numberOfIterations <
                    MAX_NUMBER_OF_ITERATIONS));
977
978             double[,] calc = new double[n, n];
979             for (int i = 0; i < n; i++)
980             {
981                 for (int j = 0; j < n; j++)
982                 {
983                     calc[i, j] = 1;
984                 }
985             }
986             foreach (KeyValuePair<string, FuzzySet> tau in taus)
987             {
988                 /*** Write tau in debug file ****/
989                 sw.Write("Tau_{" + tau.Key + "} = [");
990                 for (int i = 0; i < n; i++)
991                 {
992                     sw.Write(tau.Value[i] + " ");
993                 }
994                 sw.WriteLine("]");
995                 /*** End wirte tau ***/
996
997                 for (int i = 0; i < n; i++)
998                 {
999                     for (int j = 0; j < n; j++)
1000                    {
1001                        calc[i, j] = Math.Min(calc[i, j],
                               FuzzySet.LeftResidual(tau.Value, tau.Value)[i,
                               j]);
1002                    }
1003                }
1004            }
1005            sw.WriteLine();
1006            /*** Write Phi in debug file ****/
1007            for (int i = 0; i < n; i++)
1008            {
1009                for (int j = 0; j < n; j++)
1010                {
1011                    sw.Write(calc[i, j] + " ");
1012                }
1013                sw.WriteLine();
1014            }
1015            sw.WriteLine();
1016            /*** End wirte Phi ****/
```

```
1017
1018                sw.Close();
1019                return new FuzzyRelation() { NumberOfElements = n, Truthvalue
                        = terminal.Truthvalue, Values = calc };
1020            }
1021        }
1022    }
```

IZVORNI KÔD FAJLA A.10: Codes/FuzzyAutomata/Form1.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.IO;
7   using System.Linq;
8   using System.Text;
9   using System.Text.RegularExpressions;
10  using System.Threading.Tasks;
11  using System.Windows.Forms;
12
13  namespace FuzzyAutomata
14  {
15      public partial class Form1 : Form
16      {
17          CRLattice truthvalue = new CRLattice_Product();
18          int numberOfStates = 0;
19          FuzzySet initialStates, terminalStates;
20          Dictionary<char, FuzzyRelation> transitionRelations;
21          Factorization factorization = new FactorizationMohri();
22          FuzzyRelation phi;
23
24          private const int MAX_NUMBER_OF_STATES = 200;
25
26          public FuzzyRelation this[char key]
27          {
28              get
29              {
30                  return transitionRelations[key];
31              }
32              set
33              {
34                  transitionRelations[key] = value;
35              }
36          }
37
38          public Dictionary<char, FuzzyRelation> TransitionRelations
39          {
40              get
41              {
42                  return transitionRelations;
43              }
44          }
45
46          public Form1()
47          {
48              InitializeComponent();
49              transitionRelations = new Dictionary<char, FuzzyRelation>();
50          }
51
52          private void radioButton_CheckedChanged(object sender, EventArgs
                e)
```

```
53          {
54              if(radioButton1.Checked)
55              {
56                  truthvalue = new CRLattice_Product();
57              }
58              else if(radioButton2.Checked)
59              {
60                  truthvalue = new CRLattice_Godel();
61              }
62              else
63              {
64                  truthvalue = new CRLattice_Lukasiewicz();
65              }
66          }
67
68          private void radioButton_FactorizationChanged(object sender,
                EventArgs e)
69          {
70              if(radioButton4.Checked)
71              {
72                  factorization = new FactorizationMohri();
73              }
74              else if(radioButton5.Checked)
75              {
76                  factorization = new FactorizationTrivial();
77              }
78          }
79
80          /* Add transition matrix */
81          private void buttonTransition_Click(object sender, EventArgs e)
82          {
83              string input = Prompt.ShowDialogTwoInputs("Enter symbol",
                    "Enter fuzzy transition relation", "Fuzzy transition
                    relation?");
84              if (input.Contains('-'))
85              {
86                  string[] parts = input.Split('-');
87                  char symbol = Convert.ToChar(parts[0]);
88                  if ((parts[1][0] == '{') && (parts[1][parts[1].Length -
                        1] == '}'))
89                  {
90                      string fuzzyRelationString = parts[1].Substring(1,
                            parts[1].Length - 2);
91                      string[] fuzzyParts = fuzzyRelationString.Split(new
                            string[] { "}," }, StringSplitOptions.None);
92                      int i = 0, n = fuzzyParts.Length;
93                      double[,] result = new double[n,n];
94                      foreach (string fuzzyPart in fuzzyParts)
95                      {
96                          var fuzzySetString = (fuzzyPart[fuzzyPart.Length
                                - 1] == '}') ? fuzzyPart : fuzzyPart + "}";
97                          string valuesString = Regex.Match(fuzzySetString,
                                @"\{([^}]*)\}").Groups[1].Value;
98                          double[] convert =
                                Array.ConvertAll(valuesString.Split(','),
                                Double.Parse);
99                          for(int j = 0; j < convert.Length; j++)
100                         {
101                             result[i,j] = convert[j];
102                         }
103                         i++;
104                     }
105                     if (!transitionRelations.ContainsKey(symbol))
```

```
106                     {
107                         transitionRelations.Add(symbol, new
                                FuzzyRelation() { NumberOfElements = n,
                                Truthvalue = truthvalue, Values = result });
108                     }
109                     else
110                     {
111                         transitionRelations[symbol] = new FuzzyRelation()
                                { NumberOfElements = n, Truthvalue =
                                truthvalue, Values = result };
112                     }
113                 }
114             }
115         }
116
117         /* Add fuzzy inital state */
118         private void buttonInital_Click(object sender, EventArgs e)
119         {
120             string value = "";
121             if (initialStates != null)
122             {
123                 value = String.Join(",", initialStates.Values);
124                 value = value.Insert(0, "{");
125                 value = value.Insert(value.Length, "}");
126             }
127             string input = Prompt.ShowDialog("Enter fuzzy set of initial
                    states", "Initial states?", value);
128             if (input != null && input != "")
129             {
130                 string valuesString = Regex.Match(input,
                        @"\{([^}]*)\}").Groups[1].Value;
131                 double[] valuesDouble =
                        Array.ConvertAll(valuesString.Split(','),
                        Double.Parse);
132                 initialStates = new FuzzySet() { NumberOfElements =
                        numberOfStates = valuesDouble.Length, Truthvalue =
                        truthvalue, Values = valuesDouble };
133             }
134         }
135
136         /* Add fuzzy final state */
137         private void buttonFinal_Click(object sender, EventArgs e)
138         {
139             string value = "";
140             if (terminalStates != null)
141             {
142                 value = String.Join(",", terminalStates.Values);
143                 value = value.Insert(0, "{");
144                 value = value.Insert(value.Length, "}");
145             }
146             string input = Prompt.ShowDialog("Enter fuzzy set of terminal
                    states", "Terminal states?", value);
147             if (input != null && input != "")
148             {
149                 string valuesString = Regex.Match(input,
                        @"\{([^}]*)\}").Groups[1].Value;
150                 double[] valuesDouble =
                        Array.ConvertAll(valuesString.Split(','),
                        Double.Parse);
151                 terminalStates = new FuzzySet() { NumberOfElements =
                        numberOfStates = valuesDouble.Length, Truthvalue =
                        truthvalue, Values = valuesDouble };
152             }
```

```
153         }
154
155         private void setTruthValues()
156         {
157             initialStates.Truthvalue = truthvalue;
158             terminalStates.Truthvalue = truthvalue;
159             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
160             {
161                 transitionRelation.Value.Truthvalue = truthvalue;
162             }
163         }
164
165         /* Set fuzzy relation on a fuzzy automaton */
166         private FuzzyRelation setPhi()
167         {
168             FuzzyRelation phi =
                    FuzzyRelation.CrispEqualityRelation(initialStates.NumberOfElements,
                    truthvalue);
169             if (radioButton7.Checked)
170             {
171                 phi =
                        FuzzyRelation.GreatestWeaklyLeftInvariantFQO(initialStates,
                        transitionRelations);
172             }
173             if (radioButton8.Checked)
174             {
175                 phi =
                        FuzzyRelation.GreatestLeftInvariantFQO(initialStates,
                        transitionRelations);
176             }
177             if (radioButton9.Checked)
178             {
179                 phi =
                        FuzzyRelation.GreatestWeaklyRightInvariantFQO(terminalStates,
                        transitionRelations);
180             }
181             if (radioButton10.Checked)
182             {
183                 phi =
                        FuzzyRelation.GreatestRightInvariantFQO(terminalStates,
                        transitionRelations);
184             }
185             if (radioButton12.Checked)
186             {
187                 phi =
                        FuzzyRelation.GreatestRightInvariantFQOPR(terminalStates,
                        transitionRelations);
188             }
189             if (radioButton14.Checked)
190             {
191                 phi =
                        FuzzyRelation.GreatestRightInvariantFEPR(terminalStates,
                        transitionRelations);
192             }
193             if (radioButton13.Checked)
194             {
195                 phi =
                        FuzzyRelation.GreatestRightInvariantQOPR(terminalStates,
                        transitionRelations);
196             }
197             if (radioButton15.Checked)
198             {
```

```
199                 phi =
                        FuzzyRelation.GreatestRightInvariantEPR(terminalStates,
                        transitionRelations);
200             }
201             return phi;
202         }
203
204         /* Perform determinization by factorization of fuzzy states and
                fuzzy relation */
205         private void buttonCalculate_Click(object sender, EventArgs e)
206         {
207             setTruthValues();
208             dataGridView1.Rows.Clear();
209             dataGridView1.Refresh();
210             label1.Text = "";
211
212             phi = setPhi();
213
214             dataGridView1.ColumnCount = transitionRelations.Count + 2;
215             int i = 0;
216             dataGridView1.Columns[i++].Name = "Sigma";
217             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
218             {
219                 dataGridView1.Columns[i++].Name =
                        transitionRelation.Key.ToString();
220             }
221             dataGridView1.Columns[i++].Name = "sigma * tau";
222
223             Dictionary<string, FuzzySet> sigmas = new Dictionary<string,
                    FuzzySet>();
224             factorization.FuzzySet = initialStates * phi;
225             sigmas.Add("", factorization.f());
226
227             Queue<string> words = new Queue<string>();
228             words.Enqueue("");
229
230             int numberOfStates = 1;
231             bool proccessInterrupted = false;
232
233             while((words.Count > 0) && !proccessInterrupted)
234             {
235                 string currentWord = words.Dequeue();
236                 int index = dataGridView1.Rows.Add();
237                 DataGridViewRow row =
                        (DataGridViewRow)dataGridView1.Rows[index];
238
239                 i = 0;
240                 row.Cells[i++].Value = "sigma_{" + currentWord + "} = " +
                        sigmas[currentWord].ToString();
241                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
242                 {
243                     char symbol = transitionRelation.Key;
244                     FuzzySet sigmaNew = sigmas[currentWord] *
                            transitionRelation.Value * phi;
245                     factorization.FuzzySet = sigmaNew;
246                     double g = factorization.g();
247                     FuzzySet sigmaF = factorization.f();
248                     FuzzySet before = sigmaF;
249                     bool alreadyExists = false;
250                     foreach(KeyValuePair<string, FuzzySet> sigma in
                            sigmas)
```

```
251                     {
252                         if(sigma.Value == sigmaF)
253                         {
254                             alreadyExists = true;
255                             before = sigma.Value;
256                             break;
257                         }
258                     }
259                     if (!alreadyExists && !sigmaF.IsNull())
260                     {
261                         string newWord = currentWord + symbol;
262                         words.Enqueue(newWord);
263                         sigmas.Add(newWord, sigmaF);
264                         if (++numberOfStates > MAX_NUMBER_OF_STATES)
265                         {
266                             proccessInterrupted = true;
267                             break;
268                         }
269                     }
270                     row.Cells[i++].Value = g + " * " + before.ToString();
271                 }
272                 row.Cells[i++].Value = sigmas[currentWord] *
                        terminalStates;
273             }
274             label1.Text = "Number of states: " + numberOfStates;
275             if (proccessInterrupted)
276             {
277                 label1.Text += "\nPROCCESS WAS TERMINATED!";
278             }
279         }
280 
281         const int MAX_NUMBER_OF_ITERATIONS = 500;
282 
283         /* Compute reverse fuzzy automaton */
284         private void buttonReverse_Click(object sender, EventArgs e)
285         {
286             setTruthValues();
287             dataGridView1.Rows.Clear();
288             dataGridView1.Refresh();
289             label1.Text = "";
290 
291             phi = setPhi();
292 
293             dataGridView1.ColumnCount = transitionRelations.Count + 2;
294             int i = 0;
295             dataGridView1.Columns[i++].Name = "Tau";
296             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
297             {
298                 dataGridView1.Columns[i++].Name =
                        transitionRelation.Key.ToString();
299             }
300             dataGridView1.Columns[i++].Name = "tau * sigma";
301 
302             Dictionary<string, FuzzySet> taus = new Dictionary<string,
                    FuzzySet>();
303             factorization.FuzzySet = phi * terminalStates;
304             taus.Add("", factorization.f());
305 
306             Queue<string> words = new Queue<string>();
307             words.Enqueue("");
308 
309             int numberOfStates = 1;
```

```
310             bool proccessInterrupted = false;
311
312             while ((words.Count > 0) && !proccessInterrupted)
313             {
314                 string currentWord = words.Dequeue();
315                 int index = dataGridView1.Rows.Add();
316                 DataGridViewRow row =
                        (DataGridViewRow)dataGridView1.Rows[index];
317
318                 i = 0;
319                 row.Cells[i++].Value = "tau_{" + currentWord + "} = " +
                        taus[currentWord].ToString();
320                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
321                 {
322                     char symbol = transitionRelation.Key;
323                     FuzzySet tauNew = phi * transitionRelation.Value *
                            taus[currentWord];
324                     factorization.FuzzySet = tauNew;
325                     double g = factorization.g();
326                     FuzzySet tauF = factorization.f();
327                     FuzzySet before = tauF;
328                     bool alreadyExists = false;
329                     foreach (KeyValuePair<string, FuzzySet> tau in taus)
330                     {
331                         if (tau.Value == tauF)
332                         {
333                             alreadyExists = true;
334                             before = tau.Value;
335                             break;
336                         }
337                     }
338                     if (!alreadyExists && !tauF.IsNull())
339                     {
340                         string newWord = currentWord + symbol;
341                         words.Enqueue(newWord);
342                         taus.Add(newWord, before);
343                         if (++numberOfStates > MAX_NUMBER_OF_STATES)
344                         {
345                             proccessInterrupted = true;
346                             break;
347                         }
348                     }
349                     row.Cells[i++].Value = g + " * " + before.ToString();
350                 }
351                 row.Cells[i++].Value = taus[currentWord] * initialStates;
352             }
353             label1.Text = "Number of states: " + numberOfStates;
354             if (proccessInterrupted)
355             {
356                 label1.Text += "\nPROCCESS WAS TERMINATED!";
357             }
358         }
359
360         /* Compute Brzozowski fuzzy automaton */
361         private void buttonDReverse_Click(object sender, EventArgs e)
362         {
363             setTruthValues();
364             phi = setPhi();
365
366             // First reversal
367             Dictionary<string, FuzzySet> taus = new Dictionary<string,
                    FuzzySet>();
```

```
368             Dictionary<string, double> rests = new Dictionary<string,
                    double>();
369             Dictionary<string, double> finals = new Dictionary<string,
                    double>();
370             Dictionary<string, int> indexes = new Dictionary<string,
                    int>();
371
372             List<Tuple<char, double, string>>[] trans = new
                    List<Tuple<char, double, string>>[MAX_NUMBER_OF_STATES];
373             for (int k = 0; k < MAX_NUMBER_OF_STATES; k++)
374             {
375                 trans[k] = new List<Tuple<char, double, string>>();
376             }
377
378             factorization.FuzzySet = phi * terminalStates;
379             taus.Add("", factorization.f());
380             rests.Add("", factorization.g());
381             indexes.Add("", 0);
382
383             Queue<string> words = new Queue<string>();
384             words.Enqueue("");
385
386             int numberOfStates = 1;
387             bool proccessInterrupted = false;
388             int currentStateIndex = 0;
389
390             while ((words.Count > 0) && !proccessInterrupted)
391             {
392                 string currentWord = words.Dequeue();
393                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelations)
394                 {
395                     char symbol = transitionRelation.Key;
396                     FuzzySet tauNew = phi * transitionRelation.Value *
                            taus[currentWord];
397                     factorization.FuzzySet = tauNew;
398                     double g = factorization.g();
399                     FuzzySet tauF = factorization.f();
400                     bool alreadyExists = false;
401                     foreach (KeyValuePair<string, FuzzySet> tau in taus)
402                     {
403                         if (tau.Value == tauF)
404                         {
405                             alreadyExists = true;
406                             trans[currentStateIndex].Add(new Tuple<char,
                                    double, string>(symbol, g, tau.Key));
407                             break;
408                         }
409                     }
410                     if (!alreadyExists)
411                     {
412                         string newWord = symbol + currentWord;
413                         words.Enqueue(newWord);
414                         taus.Add(newWord, tauF);
415                         rests.Add(newWord, g);
416                         indexes.Add(newWord, numberOfStates);
417                         trans[currentStateIndex].Add(new Tuple<char,
                                double, string>(symbol, g, newWord));
418                         if (++numberOfStates > MAX_NUMBER_OF_STATES)
419                         {
420                             proccessInterrupted = true;
421                             break;
422                         }
```

```
423                     }
424                 }
425                 finals.Add(currentWord, initialStates *
                        taus[currentWord]);
426                 currentStateIndex++;
427             }
428             if (proccessInterrupted)
429             {
430                 label1.Text += "\nPROCCESS WAS TERMINATED!";
431                 return;
432             }
433
434
435             // Second reversal setup
436             FuzzySet sigmaR = new FuzzySet { NumberOfElements =
                    numberOfStates, Truthvalue = truthvalue, Values = new
                    double[numberOfStates] };
437             FuzzySet tauR = new FuzzySet { NumberOfElements =
                    numberOfStates, Truthvalue = truthvalue, Values = new
                    double[numberOfStates] };
438             sigmaR[0] = rests[""];
439             int i = 0;
440             foreach (KeyValuePair<string, double> final in finals)
441             {
442                 tauR[i++] = final.Value;
443             }
444             Dictionary<char, FuzzyRelation> transitionRelationsR = new
                    Dictionary<char, FuzzyRelation>();
445             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelations)
446             {
447                 FuzzyRelation transRev = new FuzzyRelation {
                        NumberOfElements = numberOfStates, Truthvalue =
                        truthvalue, Values = new double[numberOfStates,
                        numberOfStates] };
448                 for (i = 0; i < numberOfStates; i++)
449                 {
450                     foreach (Tuple<char, double, string> element in
                            trans[i])
451                     {
452                         if (element.Item1 == transitionRelation.Key)
453                         {
454                             transRev[i, indexes[element.Item3]] =
                                    element.Item2;
455                         }
456                     }
457                 }
458                 transitionRelationsR.Add(transitionRelation.Key,
                        transRev);
459             }
460
461             // Write reverse in debug
462             StreamWriter sw = new StreamWriter("debug.txt");
463             /*
464             for(i = 0; i < numberOfStates; i++)
465             {
466                 foreach (Tuple<char, double, string> element in trans[i])
467                 {
468                     sw.Write(i + "(" + element.Item1 + "): " +
                            element.Item2 + " => " + element.Item3 + " ");
469                 }
470                 sw.WriteLine();
471             }
```

```
472
473             foreach (KeyValuePair<string, int> idx in indexes)
474             {
475                 sw.WriteLine(idx.Key + "[" + idx.Value + "]");
476             }
477             */
478             sw.Write("sigmaR = [");
479             for (i = 0; i < numberOfStates; i++)
480             {
481                 sw.Write(sigmaR[i] + " ");
482             }
483             sw.WriteLine("]");
484
485             sw.Write("tauR = [");
486             for (i = 0; i < numberOfStates; i++)
487             {
488                 sw.Write(tauR[i] + " ");
489             }
490             sw.WriteLine("]");
491
492             foreach (KeyValuePair<char, FuzzyRelation> tr in
                    transitionRelationsR)
493             {
494                 sw.Write("delta_{0} = [", tr.Key);
495                 for (i = 0; i < numberOfStates; i++)
496                 {
497                     sw.Write("{{");
498                     for (int j = 0; j < numberOfStates; j++)
499                     {
500                         sw.Write(tr.Value[i, j] + " ");
501                     }
502                     sw.WriteLine("}}");
503                 }
504                 sw.WriteLine("]");
505             }
506             sw.Close();
507
508
509             // Second reversal
510
511             dataGridView1.Rows.Clear();
512             dataGridView1.Refresh();
513             label1.Text = "";
514
515
516             dataGridView1.ColumnCount = transitionRelationsR.Count + 3;
517             i = 0;
518             dataGridView1.Columns[i++].Name = "Initial";
519             dataGridView1.Columns[i++].Name = "Tau";
520             foreach (KeyValuePair<char, FuzzyRelation> transitionRelation
                    in transitionRelationsR)
521             {
522                 dataGridView1.Columns[i++].Name =
                        transitionRelation.Key.ToString();
523             }
524             dataGridView1.Columns[i++].Name = "Final";
525
526             Dictionary<string, FuzzySet> tausR = new Dictionary<string,
                    FuzzySet>();
527             factorization.FuzzySet = tauR;
528             tausR.Add("", factorization.f());
529
530             words = new Queue<string>();
```

```
531             words.Enqueue("");
532
533             numberOfStates = 1;
534             proccessInterrupted = false;
535
536             while ((words.Count > 0) && !proccessInterrupted)
537             {
538                 string currentWord = words.Dequeue();
539                 int index = dataGridView1.Rows.Add();
540                 DataGridViewRow row =
                        (DataGridViewRow)dataGridView1.Rows[index];
541
542                 i = 0;
543                 row.Cells[i++].Value = numberOfStates == 1 ?
                        factorization.g() : 0;
544                 row.Cells[i++].Value = "tau_{" + currentWord + "} = " +
                        tausR[currentWord].ToString();
545                 foreach (KeyValuePair<char, FuzzyRelation>
                        transitionRelation in transitionRelationsR)
546                 {
547                     char symbol = transitionRelation.Key;
548                     FuzzySet tauNew = transitionRelation.Value *
                            tausR[currentWord];
549                     factorization.FuzzySet = tauNew;
550                     double g = factorization.g();
551                     FuzzySet tauF = factorization.f();
552                     FuzzySet before = tauF;
553                     bool alreadyExists = false;
554                     foreach (KeyValuePair<string, FuzzySet> tau in tausR)
555                     {
556                         if (tau.Value == tauF)
557                         {
558                             alreadyExists = true;
559                             before = tau.Value;
560                             break;
561                         }
562                     }
563                     if (!alreadyExists && !tauF.IsNull())
564                     {
565                         string newWord = symbol + currentWord;
566                         words.Enqueue(newWord);
567                         tausR.Add(newWord, tauF);
568                         if (++numberOfStates > MAX_NUMBER_OF_STATES)
569                         {
570                             proccessInterrupted = true;
571                             break;
572                         }
573                     }
574                     row.Cells[i++].Value = g + " * " + before.ToString();
575                 }
576                 row.Cells[i++].Value = sigmaR * tausR[currentWord];
577             }
578             label1.Text = "Number of states: " + numberOfStates;
579             if (proccessInterrupted)
580             {
581                 label1.Text += "\nPROCCESS WAS TERMINATED!";
582             }
583         }
584     }
585 }
```

IZVORNI KÔD FAJLA A.11: Codes/FuzzyAutomata/Prompt.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FuzzyAutomata
{
    public static class Prompt
    {
        public static string ShowDialog(string text, string caption,
            string value)
        {
            Form prompt = new Form();
            prompt.Width = 500;
            prompt.Height = 150;
            prompt.Text = caption;
            Label textLabel = new Label() { Left = 50, Top = 20, Width =
                400, Text = text };
            TextBox textBox = new TextBox() { Left = 50, Top = 50, Width
                = 300, Text = value };
            Button confirmation = new Button() { Text = "OK", Left = 380,
                Width = 80, Top = 48 };
            confirmation.Click += (sender, e) => { prompt.Close(); };
            prompt.Controls.Add(confirmation);
            prompt.Controls.Add(textLabel);
            prompt.Controls.Add(textBox);
            prompt.ShowDialog();
            return textBox.Text;
        }

        public static string ShowDialogTwoInputs(string text1, string
            text2, string caption)
        {
            Form prompt = new Form();
            prompt.Width = 500;
            prompt.Height = 210;
            prompt.Text = caption;
            Label textLabel1 = new Label() { Left = 50, Top = 20, Width =
                400, Text = text1 };
            TextBox textBox1 = new TextBox() { Left = 50, Top = 45, Width
                = 300 };
            Label textLabel2 = new Label() { Left = 50, Top = 85, Width =
                400, Text = text2 };
            TextBox textBox2 = new TextBox() { Left = 50, Top = 110,
                Width = 300 };
            Button confirmation = new Button() { Text = "OK", Left = 380,
                Width = 80, Top = 110 };
            textBox1.LostFocus += (sender, e) =>
            {
                Form1 form1 = Application.OpenForms["Form1"] as Form1;
                char symbol = Convert.ToChar(textBox1.Text);
                if(form1.TransitionRelations.ContainsKey(symbol))
                {
                    FuzzyRelation relation = form1[symbol];
                    string output = "{";
                    for (int i = 0; i < relation.Values.GetLength(0); i++)
                    {
                        output += "{";
                        for(int j = 0; j < relation.Values.GetLength(1) -
                            1; j++)
```

```
52                          {
53                              output += relation.Values[i, j] + ",";
54                          }
55                          output += relation.Values[i,
                                relation.Values.GetLength(1) - 1];
56                          output += "}";
57                          if (i != relation.Values.GetLength(0) - 1)
58                          {
59                              output += ",";
60                          }
61                      }
62                      output += "}";
63                      textBox2.Text = output;
64                  }
65              };
66              confirmation.Click += (sender, e) => { prompt.Close(); };
67              prompt.Controls.Add(confirmation);
68              prompt.Controls.Add(textLabel1);
69              prompt.Controls.Add(textLabel2);
70              prompt.Controls.Add(textBox1);
71              prompt.Controls.Add(textBox2);
72              prompt.ShowDialog();
73              return (textBox1.Text != "" && textBox2.Text != "") ?
                    (textBox1.Text + "-" + textBox2.Text) : "";
74          }
75
76          public static int ShowDialogNumeric(string text, string caption,
                int value)
77          {
78              Form prompt = new Form();
79              prompt.Width = 300;
80              prompt.Height = 150;
81              prompt.Text = caption;
82              Label textLabel = new Label() { Left = 50, Top = 20, Width =
                    200, Text = text };
83              NumericUpDown inputBox = new NumericUpDown() { Left = 50, Top
                    = 50, Width = 100, Value = value };
84              Button confirmation = new Button() { Text = "OK", Left = 180,
                    Width = 80, Top = 48 };
85              confirmation.Click += (sender, e) => { prompt.Close(); };
86              prompt.Controls.Add(confirmation);
87              prompt.Controls.Add(textLabel);
88              prompt.Controls.Add(inputBox);
89              prompt.ShowDialog();
90              return (int)inputBox.Value;
91          }
92
93          public static double[,] ShowDialogFuzzyRelation(string text,
                string caption, int value)
94          {
95              Form prompt = new Form();
96              prompt.Width = 500;
97              prompt.Height = 250;
98              prompt.Text = caption;
99              Label textLabel = new Label() { Left = 50, Top = 20, Width =
                    200, Text = text };
100             DataGridView dataGridView = new DataGridView() { RowCount =
                    value };
101             for(int i = 0; i < value; i++)
102             {
103                 dataGridView.Columns.Add("column" + i, "a_" + (i + 1));
104             }
105
```

```
106             Button confirmation = new Button() { Text = "OK", Left = 400,
                    Width = 80, Top = 20 };
107             confirmation.Click += (sender, e) => { prompt.Close(); };
108             prompt.Controls.Add(confirmation);
109             prompt.Controls.Add(textLabel);
110             prompt.Controls.Add(dataGridView);
111             prompt.ShowDialog();
112
113             return new double[value, value];
114
115         }
116     }
117 }
```

IZVORNI KÔD FAJLA A.12: Codes/FuzzyAutomata/Program.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace FuzzyAutomata
8  {
9      static class Program
10     {
11         /// <summary>
12         /// The main entry point for the application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             Application.Run(new Form1());
20         }
21     }
22 }
```

Dodatak B

Biografija autora

Stefan P. Stanimirović rođen je 1. 9. 1989. godine u Leskovcu. Osnovnu školu "Trajko Stamenković" u Leskovcu, i Gimnaziju "Svetozar Marković" u Nišu, odeljenje za talentovane matematičare, završio je sa najvišim ocenama. Osnovne studije na Prirodno - matematičkom fakultetu Univerziteta u Nišu upisao je školske 2008/2009 godine, koje je završio 2011. godine sa prosečnom ocenom 9.96/10. Školske 2010/2011 godine bio je proglašen od strane Univerziteta u Nišu za najboljeg studenta Prirodno - matematičkog fakulteta. Školske 2011/2012 godine upisao je master akademske studije na istom fakultetu, koje je završio 2013. godine sa prosečnom ocenom 10/10. Master rad pod nazivom "Fazi relacijske nejednačine i primene" odbranio je sa najvišom ocenom.

Doktorske studije upisao je školske 2013/2014, pri čemu je položio sve ispite predviđene studijskim programom sa najvišom ocenom.

Od oktobra 2013. do aprila 2015. godine radio je u nekoliko IT kompanija na poslovima razvoja i održavanja veb aplikacija. Od aprila 2015. angažovan je kao asistent na Prirodno - matematičkom fakultetu Univerziteta u Nišu, gde izvodi vežbe na predmetima: Uvod u veb programiranje, Elektronsko izdavaštvo, Veb programiranje, Teorija informacija i kodiranje, Osnove računarstva, Uvod u objektno - orijentisano programiranje, Osnovi informatike. Kao istraživač, angažovan je takođe od aprila 2015. godine u okviru projekta "Razvoj metoda izračunavanja i procesiranja informacija: teorija i primene" (br. 174013, nosilac Prirodno-matematički fakultet u Nišu).

Od januara 2015. do juna 2016. godine angažovan je kao nastavnik u Gimnaziji "Bora Stanković" u Nišu, gde izvodi nastavu iz predmeta "Programiranje i programski jezici" u odeljenju za učenike sa posebnim sposobnostima za računarstvo i informatiku. Od septembra 2018. godine angažovan je kao nastavnik u Gimnaziji "Svetozar Marković" u Nišu, gde izvodi nastavu iz predmeta "Programiranje i programski jezici" u odeljenju za talentovane matematičare.

Učestvovao je u letnjoj školi Evropskog društva za fazi logiku septembra 2015. godine, čiji je član od iste godine. Kao autor ili koautor objavio je 11 naučnih radova, a svoje rezultate prezentovao je na naučim konferencijama: 5th International Conference Analysis, Topology, Algebra: Theory and Applications (ATA, Čačak, Srbija, jun 2016.), 7th International Conference on Algebraic Informatics (CAI, Kalamata, Grčka, jun 2017.), 9th International Workshop Weighted Automata: Theory and Applications (WATA, Leipzig, Nemačka, maj 2018.).

Bibliografija

Radovi u međunarodnim i domaćim časopisima

- S. Stanimirović, A. Stamenković, M. Ćirić, Improved algorithms for computing the greatest right and left invariant Boolean matrices and their application, Filomat, prihvaćen za štampu.
- S. Stanimirović, M. Ćirić, J. Ignjatović, Determinization of fuzzy automata by factorizations of fuzzy states and right invariant fuzzy quasi-orders, Information Sciences 469 (2018) 79–100.
- I. Micić, Z. Jančić, S. Stanimirović, Computation of the greatest right and left invariant fuzzy quasi-orders and fuzzy equivalences, Fuzzy Sets and Systems 339 (2018) 99 – 118.
- S. Stanimirović, P. Stanimirović, A. Ilić, Ballot matrix as Catalan matrix power and related identities, Discrete Applied Mathematics 160 (2012) 344–351.
- S. Stanimirović, A matrix approach to Binomial theorem, Ukrainian Mathematical Journal 64 (11) (2012) 1578–1584.
- S. Stanimirović, Some identities on Catalan numbers and hypergeometric functions via Catalan matrix power, Applied Mathematics and Computation 217 (2011) 9122–9132.
- P. Stanimirović, S. Stanimirović, Inversion of Catalan matrix plus one, Journal of Applied Mathematics and Computing 35 (2011) 497–505.
- S. Stanimirović, A generalization of the Pascal matrix and its properties, Facta Universitatis, Series Mathematics and Informatics 26 (2011) 17–27.
- P. Stanimirović, S. Stanimirović, Inverting linear combinatorial identity and generalized Catalan matrices, Linear Algebra and Its Applicatons 433 (2010) 1472–1480.
- S. Stanimirović, P. Stanimirović, M. Miladinović, A. Ilić, Catalan matrix and related combinatorial identities, Applied Mathematics and Computation 215 (2009) 796–805.

Radovi u zbornicima konferencija međunarodnog značaja

- S. Stanimirović, M. Ćirić, J. Ignjatović, An improvement of the determinization of fuzzy finite automata via factorization of fuzzy states, 7th Conference on Algebraic Informatics CAI 2017, Kalamata, Greece, 2017 (pp. 1-15).

Прилог 4/1

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	монографска
Тип записа, **ТЗ**:	текстуални / графички
Врста рада, **ВР**:	докторска дисертација
Аутор, **АУ**:	Стефан П. Станимировић
Ментор, **МН**:	Мирослав Д. Ћирић
Наслов рада, **НР**:	ПОБОЉШАНИ АЛГОРИТМИ ЗА ДЕТЕРМИНИЗАЦИЈУ ФАЗИ И ТЕЖИНСКИХ АУТОМАТА
Језик публикације, **ЈП**:	српски
Језик извода, **ЈИ**:	енглески
Земља публиковања, **ЗП**:	Србија
Уже географско подручје, **УГП**:	Србија
Година, **ГО**:	2019.
Издавач, **ИЗ**:	ауторски репринт
Место и адреса, **МА**:	Ниш, Вишеградска 33.
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)	182 стр., граф. прикази
Научна област, **НО**:	рачунарске науке
Научна дисциплина, **НД**:	теорија израчунавања
Предметна одредница/Кључне речи, **ПО**:	фази аутомати, фази језици, детерминизација
УДК	519.71, 519.713, 519.76
Чува се, **ЧУ**:	библиотека
Важна напомена, **ВН**:	
Извод, **ИЗ**:	Циљ дисертације је развој детерминизационих алгоритама базираних на концепту факторизације, као и препознавању и сажимању еквивалентних стања фази (тежинског) аутомата у конструкцији. Примењена је техника уситњења партиција за добијање побољшаних алгоритама за рачунање највећих десно и лево инваријантних Булових матрица еквиваленција и матрица квази – уређења. На крају, разматрани су начини израчунавања највећих десно и лево инваријантних фази еквиваленција и фази квази – уређења када се алгоритми за њихово рачунање, базирани на техници уситњења партиција, не завршавају у коначном броју корака.
Датум прихватања теме, **ДП**:	04.06.2018.

Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	
	Члан:	
	Члан:	
	Члан, ментор:	

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	**monograph**
Type of record, **TR**:	**textual / graphic**
Contents code, **CC**:	**doctoral dissertation**
Author, **AU**:	**Stefan P. Stanimirović**
Mentor, **MN**:	**Miroslav D. Ćirić**
Title, **TI**:	**IMPROVED ALGORITHMS FOR DETERMINIZATION OF FUZZY AND WEIGHTED AUTOMATA**
Language of text, **LT**:	**Serbian**
Language of abstract, **LA**:	**English**
Country of publication, **CP**:	**Serbia**
Locality of publication, **LP**:	**Serbia**
Publication year, **PY**:	**2019**
Publisher, **PB**:	**author's reprint**
Publication place, **PP**:	**Niš, Višegradska 33.**
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	**182 p. ; graphic representations**
Scientific field, **SF**:	**computer science**
Scientific discipline, **SD**:	**theory of computing**
Subject/Key words, **S/KW**:	**fuzzy automata, fuzzy languages, determinization**
UC	**519.71, 519.713, 519.76**
Holding data, **HD**:	**library**
Note, **N**:	
Abstract, **AB**:	The aim of this dissertation is the development of determinization algorithms based on the concept of factorizations, as well as computing and merging of the indistinguishable states of fuzzy (weighted) automaton under construction. We apply the partition refinement technique to obtain improved algorithms for computing the greatest right and left invariant Boolean equivalence and quasi – order matrices. In the end, we consider ways to compute the greatest right and left invariant fuzzy equivalences and fuzzy quasi – orders when the algorithms for their computation, based on the partition refinement technique, are unable to stop in a finite number of steps.
Accepted by the Scientific Board on, **ASB**:	04.06.2018.
Defended on, **DE**:	

Defended Board, **DB**:	President:	
	Member:	
	Member:	
	Member:	
	Member, Mentor:	

ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

ПОБОЉШАНИ АЛГОРИТМИ ЗА ДЕТЕРМИНИЗАЦИЈУ ФАЗИ И ТЕЖИНСКИХ АУТОМАТА

која је одбрањена на Природно – математичком факултету Универзитета у Нишу:

- резултат сопственог истраживачког рада;
- да ову дисертацију, ни у целини, нити у деловима, нисам пријављивао/ла на другим факултетима, нити универзитетима;
- да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица.

Дозвољавам да се објаве моји лични подаци, који су у вези са ауторством и добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада, и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, 11.4.2019.

Потпис аутора дисертације:

Стефан Станимировић

Стефан П. Станимировић

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНОГ И ЕЛЕКТРОНСКОГ ОБЛИКА ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Наслов дисертације:

ПОБОЉШАНИ АЛГОРИТМИ ЗА ДЕТЕРМИНИЗАЦИЈУ ФАЗИ И ТЕЖИНСКИХ АУТОМАТА

Изјављујем да је електронски облик моје докторске дисертације, коју сам предао/ла за уношење у **Дигитални репозиторијум Универзитета у Нишу,** истоветан штампаном облику.

У Нишу, 11.4.2019.

Потпис аутора дисертације:

Стефан П. Станимировић

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да у Дигитални репозиторијум Универзитета у Нишу унесе моју докторску дисертацију, под насловом:

ПОБОЉШАНИ АЛГОРИТМИ ЗА ДЕТЕРМИНИЗАЦИЈУ ФАЗИ И ТЕЖИНСКИХ АУТОМАТА

Дисертацију са свим прилозима предао/ла сам у електронском облику, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио/ла.

1. Ауторство **(CC BY)**
2. Ауторство – некомерцијално **(CC BY-NC)**
3. **Ауторство – некомерцијално – без прераде (CC BY-NC-ND)**
4. Ауторство – некомерцијално – делити под истим условима **(CC BY-NC-SA)**
5. Ауторство – без прераде **(CC BY-ND)**
6. Ауторство – делити под истим условима **(CC BY-SA)**

У Нишу, 11.4.2019.

Потпис аутора дисертације:

Стефан Станимировић

Стефан П. Станимировић