Datorikas fakultāte

Galīgu automātu pārejas funkcijas sarežģītība

Māris Valdats

Promocijas darbs datorzinātnē

darba zinātniskais vadītājs
prof. Juris Smotrovs

2019. gada 6. februārī

Anotācija

Šajā darbā no sarežģītības viedokļa tiek aplūkots galīgu automātu stāvokļu kodēšanas uzdevums. Tiek ieviests un pētīts galīgu automātu un regulāru valodu sarežģītības mērs, BC-sarežģītība, kas pēc būtības ir galīga automāta pārejas funkcijas loģiskās shēmas sarežģītība.

Regulāru valodu BC-sarežģītība ir novērtēta attiecībā pret stāvokļu sarežģītību un tai tiek iegūtas sakrītošas augšējās un apakšējās robežas gandrīz visām valodām ar doto stāvokļu sarežģītību. Šādas robežas tiek iegūtas arī attiecībā pret nedeterminēto stāvokļu sarežģītību, kur gan tās nav sakrītošas. Tiek pierādīts, ka galīgu automātu minimizācija var novest pie vairāk nekā polinomiāla to BC-sarežģītības pieauguma.

Galīgi automāti, kas izteikti ar loģisko elementu shēmu, tiek aplūkoti arī no algoritmiskās sarežģītības viedokļa. Tiek pamatots, ka daudzi vienkārši jautājumi (stāvokļu sasniedzamība vai ekvivalence, automāta minimizācija) šādā reprezentācijā ir PSPACE-pilnīgi.

Saturs

1. nodaļa

Ievads

Galīgu automātu stāvokļa sarežģītība [38][19] tiek pētīta jau kopš automātu teorijas pirmsākumiem, vairāk neka 50 gadus, un kopš tā laika tā ir bijusi galvenais mērs, lai novērtētu galīga automāta sarežģītību. Bet, lai arī tā kalpo labi automātiem standarta reprezentācijā, tā ne vienmēr pilnībā atspoguļo automāta ”intuitīvo” sarežģītību. Lai to ilustrētu, aplūkosim divus galīgus automātus ar vienu un to pašu stāvokļu skaitu, no kuriem viens ir ”intuitīvi” krietni sarežģītāks par otru.

Aplūkosim galīgu determinētu automātu (GDA), kas pazīst tādu regulāru valodu binārā alfabētā, kura sastāv no vārdiem, kuros starp pēdējiem 1000 simboliem ir pāra skaits vieninieku. Viegli pierādīt, ka šādam automātam nepieciešami 2^{1000} stāvokļi (un ar to pietiek), bet šādu GDA var viegli realizēt, glabājot tā stāvokļu telpu 1000 bitu reģistrā, kurš atceras pēdējos 1000 simbolus.

No otras puses, aplūkosim ”patvaļīgu” GDA binārā alfabētā ar 2^{1000} ieejas stāvokļiem. Nav vispārīgas metodes, kā to aprakstīt citādi, kā ar tā stāvokļu pārejas tabulu, kura sastāvēs no 2^{1001} rindiņām un kura (kā tas tiek uzskatīts) būtu lielāka par mūsu visumu.

Lielu stāvokļu skaitu var viegli attēlot kompaktā formā: 2^n stāvokļus var iekodēt n stāvokļa bitos. Tas ir spēkā abiem iepriekš aplūkotajiem GDA. Bet transformācija, kura jāizrēķina, pārejot uz nākamo stāvokli, dažos gadījumos var būt vienkārša, bet citos ļoti sarežģīta.

Stāvokļu kodēšanas uzdevums: atrast galīgam automātam efektīvu veidu, kā nokodēt tā stāvokļus, lai tā pārejas funkcija būtu ”vienkārša”, arī tiek pētīts jau kopš pašiem automātu teorijas pirmsākumiem[17]. Pie tam šī ”vienkāršība” var tikt saprasta dažādi: klasiskajā gadījumā[9][24] stāvokļu kodējumā mēģina minimizēt atkarības starp mainīgajiem, tādā veidā samazinot realizējamās Būla funkcijas sarežģītību, bet daudzi raksti[5][22] ir veltīti arī efektīva stāvokļu kodējuma atrašanai, lai minimizētu vidējo pārslēgšanos skaitu atmiņas elementiem (bitiem), tādējādi minimizējot vidējo elektrības patēriņu.

Taču, lai gan stāvokļu kodēšanas uzdevums ir pētīts jau ilgāk nekā 50 gadus, tas līdz šim ir aplūkots tikai kā optimizācijas uzdevums. Kā ilustrāciju šim faktam var minēt to, ka daudzos rakstos tā mērķis tiek saukts par ”iegūtās loģiskās shēmas laukuma minimizāciju”[13], kas norāda uz šo pētījumu praktisko raksturu. Bet pēc savas būtības stāvokļu kodēšanas uzdevums ir tipisks sarežģītības uzdevums.

Tā mēs nonākam pie šī darba galvenās tēmas — BC-sarežģītības, kas ir sarežģītības mērs galīgiem automātiem un regulārām valodām. Pavisam vienkārši izsakoties, BC-sarežģītība ir GDA pārejas funkcijas izteiktas kā loģisko elementu shēmas sarežģītība. Precīzāk BC-sarežģītība tiks nodefinēta šī darba 4. nodaļā.

Pirms tam, 2. nodaļā, tiks aplūkoti nepieciešamie matemātikas un datorzinātnes pamatjēdzieni, bet 3. nodaļā tiks precīzi nodefinēts, kā galīgu automātu aprakstīt ar loģisko elementu shēmu (shēmām).

Tālāk, 4. nodaļas turpinājumā, BC-sarežģītība tiek salīdzināta ar stāvokļu sarežģītību, tiek pamatots, ka GDA (un regulārām valodām) ar vienu un to pašu stāvokļu sarežģītību, BC-sarežģītība var atšķirties eksponenciāli. Regulārām valodām ar dotu stāvokļa sarežģītību tiek pierādīts "Šenona efekts": gandrīz visiem automātiem BC-sarežģītība ir tuva savai maksimālajai iespējamajai vērtībai.

Nākamajā, 5. nodaļā BC-sarežģītība tiek salīdzināta ar nedeterminēto stāvokļu sarežģītību. Līdzīgi kā iepriekš, tiek iegūtas regulāru valodu BC-sarežģītības augšējās un apakšējās robežas attiecībā pret to nedeterminēto stāvokļu sarežģītību. Lai arī Šenona efekts nedeterminētās stāvokļu sarežģītības gadījumā netiek pierādīts, augšējā un apakšējā robežas ir tuvas, tās atšķiras ne vairāk kā 4 reizes.

BC-sarežģītība dažādām valodu operācijām: apvienojumam, šķēlumam, konkatenācijai, apvēršanai un Klīni slēgumam ir aplūkota 6. nodaļā. Var novērot, ka BC sarežģītība valodai, kas iegūta jebkuras aplūkotās operācijas rezultātā, ir mazāka, nekā tā būtu patvaļīgai valodai ar doto stāvokļu skaitu.

BC-sarežģītības saistība ar GDA minimizāciju aplūkota 7. nodaļā. Tur atrodams viens no šī darba galvenajiem rezultātiem: teorēma, kurā pierādīts, ka GDA stāvokļu skaita minimizācija dažos gadījumos var novest pie vairāk nekā polinomiāla BC-sarežģītības pieauguma.

Tā kā BC-sarežģītība ir cieši saistīta ar stāvokļu kodēšanas uzdevumu un tas parasti tiek aplūkots galīgiem automātiem ar izeju, tad 8. nodaļā aplūkota automātu ar izeju loģiskās shēmas reprezentācijas un BC-sarežģītība.

Darba 9. nodaļā aplūkota algoritmiskā sarežģītība dažādām darbībām ar automātiem to loģiskās shēmas reprezentācijā. Pamatots, ka daudzas operācijas, kas GDA standarta reprezentācijā ir salīdzinoši "vieglas" (stāvokļu sasniedzamība, GDA ekvivalence un minimizācija), loģiskās shēmas reprezentācijā ir PSPACE-pilnīgas.

Darba nobeigumā aplūkoti BC-sarežģītībai līdzīgi jēdzieni (Kolmogorova sarežģītība, alternējoši automāti, regulāru valodu loģiskās shēmas sarežģītība) un paskaidrotas būtiskākās atšķirības.

2. nodaļa

Izmantotie jēdzieni un apzīmējumi

2.1. Vārdi un valodas

Par *vārdu* kāda alfabētā Σ sauc galīgu simbolu virkni $w = w_1 w_2 \dots w_k$, kur $w_i \in \Sigma$ visiem i, $k = |w|$ šajā gadījumā ir vārda garums. Ar Σ^* apzīmē visu vārdu kopu alfabētā Σ.

Par vārdu $u = u_1 u_2 \dots u_k$ un $v = v_1 v_2 \dots v_l$ *konkatenāciju* sauc vārdu

$$uv = u_1 u_2 \dots u_k v_1 v_2 \dots v_l.$$

Ar u^n apzīmēsim n vienādu vārdu u konkatenāciju, ar $u^0 = \varepsilon$ apzīmēsim tukšo vārdu, kurš nesatur nevienu simbolu, jebkuram vārdam u ir spēkā, ka $u = \varepsilon u = u\varepsilon$. Nešķirosim alfabēta simbolus no vārdiem ar garumu 1. Par vārda $u = u_1 u_2 \dots u_k$ *apvērsto vārdu* u^R sauksim u uzrakstītu no otra gala $u^R = u_k \dots u_2 u_1$.

Par *valodu* sauc patvaļīgu vārdu kopu, tas ir Σ^* apakškopu. Par valodu apvienojumu un šķēlumu sauc to apvienojumu un šķēlumu kopu teorijas izpratnē. Par valodas L *apvērsto valodu* L^R sauc visu valodas L apvērsto vārdu kopu $u \in L \iff u^R \in L^R$. Par valodu L_1 un L_2 konkatenāciju $L_1 L_2$ sauc visu vārdu uv kopu, tādu, ka $u \in L_1$ un $v \in L_2$. Apzīmēsim $L^1 = L$, $L^2 = LL$, $L^3 = L^2 L$ utt, papildus tam $L^0 = \{\varepsilon\}$. Tad par *Klīni slēgumu* valodai L sauc valodu

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \dots = \bigcup_{i=0}^{+\infty} L^i.$$

Par *regulārām valodām* alfabētā Σ sauc tukšo valodu, valodu, kas sastāv no tukšā vārda, valodas, kas sastāv no jebkura viena burta vārda, kā arī visas valodas, ko var iegūt no iepriekšminētajām, atkārtoti pielietojot apvienojumu, konkatenāciju un Klīni slēgumu.

2.2. Galīgie automāti un regulāras valodas

2.2.1. Galīgi determinēti automāti

Galīgs determinēts automāts (GDA) [29] ir viens no populārākajiem skaitļošanas modeļiem, kas datorzinātnē tiek pētīts jau kopš 50. gadiem. Tam ir vairākas variācijas, bet mēs galvenokārt aplūkosim GDA bez izejas (akceptorus) un tikai 8. nodaļā nedaudz pievērsīsimies galīgiem automātiem ar izejas lenti.

Definīcija Galīgs determinēts automāts (GDA) ir kortežs $(Q, \Sigma, \delta, q_0, \tilde{Q})$, kur

1. Q ir stāvokļu telpa (galīga kopa)
2. Σ ir ieejas alfabēts (galīga kopa)
3. $\delta : \Sigma \times Q \to Q$ ir pārejas funkcija
4. $q_0 \in Q$ ir sākuma stāvoklis
5. $\tilde{Q} \subseteq Q$ ir akceptējošo (beigu) stāvokļu kopa

GDA sāk darbu stāvoklī q_0, katrā solī tas nolasa no ieejas lentes vienu simbolu $x \in \Sigma$ un nomaina savu stāvokli. Ja GDA atrodas stāvoklī $q \in Q$ un nolasa ieejas simbolu $x \in \Sigma$, tad tas pāriet uz jaunu stāvokli $\delta(x, q)$.

Pārejas funkciju δ dabīgā veidā (ar matemātisko indukciju) var paplašināt tā, lai tā kā pirmo argumentu saņemtu Σ^* — visu iespējamo vārdu kopu alfabētā Σ. Ja ar ε apzīmē tukšo vārdu, tad $\delta(\varepsilon, q) = q$ un jebkuram vārdam $u \in \Sigma^*$ un simbolam $x \in \Sigma$ izpildās $\delta(ux, q) = \delta(x, \delta(u, q))$.

Ja GDA pēc ieejas vārda nolasīšanas atrodas kādā stāvoklī $q \in \tilde{Q}$, tad tas šo vārdu akceptē, pretējā gadījumā noraida. Saka, ka GDA A pazīst valodu L, ja tas akceptē visus vārdus, kas pieder šai valodai, un noraida visus vārdus, kas tai nepieder:

$$\omega \in L \iff \delta(\omega, q_0) \in \tilde{Q}.$$

Ir zināms, ka ar GDA var akceptēt visas regulārās valodas un nekādas citas (Klīni teorēma). Par regulāras valodas L *stāvokļu sarežģītību* $sc(L)$ sauc mazāko iespējamo stāvokļu skaitu GDA, kas pazīst šo valodu.

Divus GDA sauc par *ekvivalentiem*, ja tie pazīst vienu un to pašu valodu. Katram GDA A var atrast *minimālo* ekvivalento GDA $M(A)$ (GDA ar minimālo iespējamo stāvokļu skaitu, kas ekvivalents dotajam GDA), šāds minimālais GDA $M(A)$ katram GDA A ir tieši viens (ar precizitāti līdz izomorfismam). Tāpat ar $M(L)$ apzīmēsim minimālo GDA, kas pazīst doto regulāro valodu L.

Minimālā GDA atrašanu sauc par GDA *minimizēšanu*, tai efektīvs algoritms (minimizācijas algoritms) [38], kas darbojas polinomiālā laikā no GDA stāvokļu skaita. Lielos vilcienos minimizācijas process sastāv no divām daļām:

- vispirms tiek atmesti visi nesasniedzamie GDA stāvokļi (tie, kuros GDA nevar nonākt ne ar kādu ieejas vārdu)
- pēc tam tiek "salīmēti" kopā visi ekvivalentie stāvokļi (tie, kurus uzstādot par GDA sākuma stāvokļiem, tas akceptētu vienu un to pašu valodu).

Šajā darbā mēs aplūkosim tikai pilnus automātus, kam pārejas funkcija definēta visam $\Sigma \times Q$ vērtībām. Nepilnus automātus (kam šī funkcija dažām vērtībām var nebūt definēta) var pārveidot par pilniem, pieliekot klāt ne vairāk ka vienu stāvokli.

2.2.2. Galīgi nedeterminēti automāti

Par galīgu nedeterminētu automātu GNA sauc kortežu $(Q, \Sigma, \delta, q_0, \tilde{Q})$, kuram

1. Q ir stāvokļu telpa (galīga kopa)
2. Σ ir ieejas alfabēts (galīga kopa)
3. $\delta : \Sigma \times Q \to 2^Q$ ir pārejas funkcija
4. $Q_0 \subseteq Q$ ir sākuma stāvokļu kopa
5. $\tilde{Q} \subseteq Q$ ir akceptējošo (beigu) stāvokļu kopa

GNA ir GDA vispārinājums. Atšķirībā no GDA , nolasot ieejas simbolu, GNA no katra stāvokļa var pāriet uz vairākiem citiem stāvokļiem. Sākot darbu, tas atrodas stāvokļu kopā Q_0 un, pēc katra simbola nolasīšanas, tas pāriet uz kādā citu savu stāvokļu apakškopu. Ja tas atrodas stāvokļu kopā $S \subseteq Q$ un nolasa ieejas simbolu x, tad tas pāriet uz stāvokļu kopu S', kurai

$$S' = \bigcup_{q \in S} \delta(q, x).$$

Ja pēc visa vārda nolasīšanas GNA atrodas stāvokļu kopā S un kāds no šiem stāvokļiem ir beigu stāvoklis ($S \cap \tilde{Q} \neq \emptyset$), tad tas šo vārdu akceptē. GNA pazīst valodu L, ja tas akceptē visus vārdus, kas pieder šai valodai, un noraida visus vārdus, kas nepieder šai valodai. Vienkāršības pēc aplūkosim tikai GNA bez ε-pārejām.

GNA ekvivalenta GDA konstrukcija aprakstīta jau 1959 gadā[32], kur arī dots augšējais novērtējums šāda GDA izmēram — tas ir 2^n, ja sākotnējajam GNA bija n stāvokļu. Vēlāk tika pamatots[30], ka šis novērtējums ir precīzs un dažām valodām atbilstošie minimālie GDA ir eksponenciāli lielāki par minimālajiem GNA.

Par regulāras valodas L *nedeterminēto stāvokļu sarežģītību* $nsc(L)$ sauc minimālo stāvokļu skaitu, kas ir kādam GNA, kas pazīst šo valodu. No iepriekšminētā izriet, ka jebkurai regulārai valodai L ir spēkā

$$ncs(L) \leq sc(L) \leq 2^{nsc(L)}.$$

2.2.3. GDA un GNA skaita novērtējums

Dažādo GDA skaits ar s stāvokļiem k simbolu alfabētā ir $2^s s^{ks}$ — iespējamas 2^s beigu stāvokļu kopas un s^{ks} pārejas funkcijas. Taču daudzi no tiem ir ekvivalenti un pat izomorfi. Šajā darbā mums būs nepieciešams neekvivalentu GDA un GNA ar ne vairāk ka s stāvokļiem skaita apakšējais novērtējums, kuru ņemsim no raksta [14] nedaudz vienkāršotā formā.

Apzīmēsim ar $\mathfrak{L}_n^k$ ($\mathfrak{N}_n^k$) dažādu regulāru valodu skaitu k simbolu alfabētā, kuru stāvokļu (nedeterminētā stāvokļu) sarežģītība nepārsniedz n. Tādā gadījumā

2.1. Teorēma ([14])

$$\begin{array}{llll} |\mathfrak{L}_s^k| \geq s^{(k-1)s} & \text{un} & 2^{(k-1)s^2} \leq |\mathfrak{N}_s^k| \leq 2s2^{ks^2}, & \text{ja } k \geq 2, \\ |\mathfrak{L}_s^1| \geq 2^s & \text{un} & 2^s \leq |\mathfrak{N}_s^1| \leq 2^{s \log s}, & \text{ja } k = 1. \end{array}$$

Šeit un turpmāk ar log apzīmēsim logaritmu pie bāzes 2.

2.3. Asimptotiskie novērtējumi

Lai asimptotiski salīdzinātu divas funkcijas f un g, mēs izmantosim sekojošu apzīmējumu (kas ņemts no [28]):

$$f(n) \lesssim g(n) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} \le 1$$

Šis apzīmējums ir precīzāks nekā standarta apzīmējums $f(x) = O(g(x))$, ko lieto sarežģītības teorijā, jo tas ņem vērā konstantus reizinātājus. Ar standarta apzīmējumiem to var izteikt šādā veidā:

$$f(n) \lesssim g(n) \iff f(n) < g(n)(1 + o(1)).$$

Galīgu kopu saimei $\{S_n\}$ teiksim, ka īpašība $P(x)$ izpildās *gandrīz visiem* $x \in S_n$, ja to x daļa, kuriem $P(x)$ neizpildās, tiecas uz nulli, kad n tiecas uz bezgalību:

$$P(x) \text{ gandrīz visiem } x \in S_n \iff \lim_{n\to\infty} \frac{|\{x \in S_n : \neg P(x)\}|}{|S_n|} = 0.$$

Apvienojot šos abus jēdzienus, teiksim, ka $f(x) \lesssim h(n)$ gandrīz visiem $x \in S_n$, ja eksistē tāda funkcija $g(n)$, ka $f(x) \le g(n)$ gandrīz visiem $x \in S_n$ un $g(n) \lesssim h(n)$.

Teiksim, ka summā $f(n) + g(n)$ saskaitāmais $f(n)$ dominē pār $g(n)$, ja

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0,$$

un bieži izmantosim to, ka šādā gadījumā $f(n) + g(n) \lesssim f(n)$.

2.4. Loģisko elementu shēmas

Mēs aplūkosim loģisko elementu shēmas (loģiskās shēmas) to parastajā izpratnē, izmantojot tikai standarta bāzes elementus (&, $\vee$, $\neg$).

Loģisko elementu shēma ir orientēts grafs bez cikliem, kura katrā virsotnē ienāk ne vairāk kā divas šķautnes. Ja virsotnē neienāk neviena šķautne, tad šo virsotni sauc par ieeju un tai piekārto ieejas mainīgo x_i vai konstanti 0 vai 1. Pārējās virsotnes sauc par loģiskajiem elementiem un apzīmē ar Būla funkcijām:

1. $\neg$, ja virsotnē ienāk viena šķautne,

2. & vai $\vee$, ja virsotnē ienāk divas šķautnes.

Dažas no virsotnēm tiek uzskatītas par izejas virsotnēm un apzīmētas ar simboliem $y_1, \dots, y_m$. Attēlos uzskatāmības dēļ izejas virsotnes mēs attēlosim kā atsevišķas virsotnes bez izejām ar vienu ieeju, kurā ienāks bultiņa no "īstās" izejas virsotnes.

Loģisko elementu shēma ar n ieejas virsotnēm un m izejas virsotnēm dabīgā veidā rēķina kādu Būla funkciju $F : \{0,1\}^n \to \{0,1\}^m$. Katrai virsotnei a induktīvi pēc garāka iespējamā ceļa garuma no kādas ieejas virsotnes piekārtosim Būla funkciju $f_a : \{0,1\}^n \to \{0,1\}$. Katrai ieejas virsotnei x_i šī funkcija ir $f(x_1, \dots, x_n) = x_i$, pārejām virsotnēm tā ir atbilstošā Būla funkcija (&, $\vee$, $\neg$) no atbilstošo ieejas virsotņu (virsotnes) vērtībām. Visas loģisko elementu shēmas rēķinātā funkcija $F : \{0,1\}^n \to \{0,1\}^m$ ir:

$$F = f_{y_1} \times f_{y_2} \times \cdots \times f_{y_m}.$$

Par loģiskās shēmas F sarežģītību $C(F)$ sauksim tās loģisko elementu skaitu.

Katru Būla funkciju $f : \{0,1\}^n \to \{0,1\}^m$ var aprakstīt ar loģisko shēmu (bezgalīgi) daudzos veidos. Par funkcijas sarežģītību $C(f)$ sauksim minimālo sarežģītību kāda ir kādai loģisko shēmai, kura apraksta šo Būla funkciju.

Apzīmēsim ar $N(n, m, c)$ to Būla funkciju $\{0,1\}^n \to \{0,1\}^m$ skaitu, kuru sarežģītība nepārsniedz c. Šo skaitli var novērtēt no augšas, novērtējot cik ir dažādu loģisko shēmu ar c elementiem.

2.2. Teorēma ([28])

$$N(n, m, c) < 9^{c+n}(c+n)^{c+m}.$$

Pierādījums Novērtēsim loģisko shēmu skaitu ar n ieejām, m izejām un c elementiem. Piešķirsim numurus no 1 līdz c visiem loģiskajiem elementiem, un no $c+1$ līdz $c+n$ visām ieejām. Katram loģiskajam elementam piekārtosim skaitļu trijnieku (i_1, i_2, t), kur

- $1 \le i_1, i_2 \le c+n$ ir loģiskā elementa pirmās un otrās ieejas numurs (loģiskā elementa ieejā var tikt padots vai nu mainīgais no loģiskās shēmas ieejas, vai arī kāds cits šīs loģiskās shēmas elements).
- $1 \le t \le 3$ ir attiecīgā loģiskā elementa tips (&, $\vee$, $\neg$).

Negācijas ($\neg$) gadījumā, kam ir tikai viens ieejas arguments, varam uzskatīt, ka $i_1 = i_2$. Katru no m izejas elementiem var aprakstīt ar tam atbilstošā loģiskā elementa (vai ieejas bita) numuru. Tātad šādu loģisko shēmu kopumā ir ne vairāk kā $(3(c+n)^2)^c \cdot (c+n)^m$. Pie tam katru loģisko shēmu mēs esam ieskaitījuši $c!$ reizes visām dažādajām c loģisko elementu numerācijām. Tāpēc

$$N(n, m, c) < \frac{(3(c+n)^2)^c (c+n)^m}{c!}.$$

Novērtējot $c! \ge \frac{c^c}{3^c}$, iegūst

$$N(n, m, c) < 3^c 3^c (c+n)^c (c+n)^m \frac{(c+n)^c}{c^c},$$

un, tā kā

$$\frac{(c+n)^c}{c^c} = ((1 + \frac{n}{c})^{\frac{c}{n}})^n < 3^n,$$

tad

$$N(n, m, c) < 9^c (c+n)^c (c+n)^m 3^n < 9^{c+n}(c+n)^{c+m}. \quad \blacksquare$$

2.4.1. Šenona efekts un Šenona funkcija

Pirms 70 gadiem ar savu slaveno rakstu [35] Klods Šenons lika pamatus loģisko elementu shēmu un Būla funkciju sarežģītībai. Un jau tad viņš ievēroja, ka gandrīz visām Būla funkcijām to sarežģītība ir tuvu maksimāli iespējamajai vērtībai. Lai arī pilnībā šo efektu viņš nepierādīja un tas tika noslīpēts tikai tālākos darbos [28], tomēr, tā kā viņš bija pirmais, kurš ar nekonstruktīvām metodēm ieguva apakšējo robežu Būla funkciju sarežģītībai gandrīz visām funkcijām, šo efektu tagad sauc viņa vārdā.

Vispārīgā gadījumā, kādai objektu klasei F_n un to sarežģītības mēram $C()$ ir spēkā *Šenona efekts*, ja gandrīz visiem $x \in F_n$ ir spēkā

$$C(x) \gtrsim \max\{C(y) : y \in F_n\}.$$

Tā kā turpmākajā darbā šo jēdzienu mēs lietosim diezgan bieži, tad aplūkosim šeit, kā tad tas izpaužas klasiskajā gadījumā, kad objekti ir Būla funkcijas un sarežģītība ir to loģiskās shēmas sarežģītība.

2.3. Teorēma ([28]) *Visām Būla funkcijām* $f : \{0,1\}^n \to \{0,1\}^m$

$$C(f) \lesssim \frac{m \cdot 2^n}{n + \log m},$$

Gandrīz visām Būla funkcijām $f : \{0,1\}^n \to \{0,1\}^m$

$$C(f) > \frac{m \cdot 2^n}{n + \log m}.$$

Pierādījums Šīs teorēmas (vai precīzāk, divu atsevišķu teorēmu) pierādījums gadījumam kad $m = 1$ ir atrodams daudzās grāmatās [43], gadījumu, kad $m > 1$, atrast ir grūtāk (krievu valodā tas ir [28]), bet metode ir tā pati kā $m = 1$ gadījumā.

Augšējo robežu iegūst ar speciālas konstrukcijas palīdzību, bet apakšējo robežu — nekonstruktīvi, pēc Dirihlē principa (saskaitot visas Būla funkcijas no n mainīgajiem(to ir 2^{m2^n}), saskaitot visas loģisko elementu shēmas no n mainīgajiem ar m izejas mainīgajiem un ne vairāk kā $\frac{m2^n}{n+\log m}$ loģiskajiem elementiem (ar 2.2 teorēmas palīdzību) un novērtējot, ka pirmais skaitlis ir daudz lielāks par otro).

Dažos gadījumos mums nāksies saskarties ar Būla funkcijām, kuras nav definētas visām iespējamajām savu argumentu vērtībām. Mums noderēs šāds 2.3. teorēmas augšējās robežas vispārinājums. Apzīmēsim ar $F^{\nu,m}$ Būla funkciju ar n ieejas mainīgajiem un m izejas mainīgajiem klasi, kuras var pieņemt dažādas vērtības leksikogrāfiski pirmajām ν argumentu vērtībām, bet pārējām $2^n - \nu$ argumentu vērtībām, to vērtība ir 0^m.

2.4. Teorēma ([28]) *Ja* (ν_i, m_i) *ir tāda skaitļu pāru virkne, ka*

- $\nu_i \to +\infty$
- $\frac{\log m_i}{\nu_i} \to +\infty$

tad visiem $x \in C(F^{\nu_i,m_i})$

$$C(x) \lesssim \frac{\nu_i m_i}{\log(\nu_i m_i)}$$

Gadījumā, ja visi $\nu_i = 2^{n_i}$ ir divnieka pakāpes, mēs iegūsim tieši to pašu rezultātu, ko no 2.3. teorēmas augšējās robežas.

Iepriekšējo teorēmu nekonstruktīvie apakšējie novērtējumi stipri kontrastē ar labākajiem konstruktīvajiem novērtējumiem konkrētām funkcijām. Piemēram, $m =$

1 gadījumā gandrīz visām Būla funkcijām $C(f) > \frac{2^n}{n}$, bet labākie apakšējie novērtējumi konkrētām funkcijām vispārīgajā gadījumā ir lineāri attiecībā pret n.

Tas ir kaut kādā ziņā pārsteidzošs fakts, ka, lai arī zināms, ka gandrīz visas funkcijas ir "sarežģītas", tomēr atrast kādu konkrētu "sarežģītu" funkciju nav iespējams. Neskatoties uz to reizēm kāda konkrēta "sarežģīta" funkcija ir vajadzīga un tāpēc lieto Šenona funkcijas jēdzienu.

Par *Šenona funkciju* no n mainīgajiem Sh_n sauc leksikogrāfiski pirmo Būla funkciju $f : \{0,1\}^n \to \{0,1\}$, kuras sarežģītība ir maksimālā iespējamā no visam šādam funkcijām. No 2.3. teorēmas izriet, ka Šenona funkcijas sarežģītība ir lielāka par $2^n/n$.

2.4.2. Dažu "praktisku" Būla funkciju sarežģītība

Lai arī iepriekš tika parādīts, ka lielākajai daļai Būla funkciju sarežģītība ir tuvu maksimālajai un tā ir eksponenciāla attiecībā pret ieejas bitu skaitu, tomēr lielai daļai "praktisku" funkciju šī sarežģītība ir daudz mazāka. Sekojošā teorēmā aplūkosim sarežģītību 4 pamatfunkcijām: konstantes pieskaitīšanai, salīdzināšanai ar konstanti, izvēles funkcijai un moduļa funkcijai, kuras vēlāk tiks izmantotas kā sastāvdaļas (bloki) no kurām tiks būvētas sarežģītākas loģiskās shēmas.

2.5. Teorēma *Sarežģītība Būla funkcijai $f_u : \{0,1\}^n \to \{0,1\}$, kurai*

$$f_u(x) = 1 \leftrightarrow x > u,$$

nav lielāka par n.

Sarežģītība Būla funkcijai $f_v : \{0,1\}^n \to \{0,1\}^n$, kurai

$$f_v(x) = x \pm v \mod 2^n,$$

nav lielāka par $6n$

Sarežģītība Būla funkcijai $f : \{0,1\}^{2n+1} \to \{0,1\}^n$, kurai

$$f(x,y,t) = \begin{cases} x, & \textit{ja } t = 0 \\ y, & \textit{ja } t = 1, \end{cases}$$

nav lielāka par $3n+1$ (x un y ir n-bitu mainīgie, t ir viena bita mainīgais).

Sarežģītība Būla funkcijai $f_{u,v} : \{0,1\}^n \to \{0,1\}$, kurai

$$f_{u,v}(x) = 1 \leftrightarrow x = u \mod v,$$

nav lielāka par $15n^2$.

Pierādījums Sāksim ar salīdzināšanas funkciju. Tai mēs varam uzkonstruēt loģisko shēmu, izmantojot parasto atņemšanu $u - x$ ar pārnesumu. Ja pēc pēdējā (n-tā) bita pārnesuma vērtība būs 1, tad x ir mazāks par u. Apzīmēsim ieejas mainīgos (bitus) ar $x_n \dots x_2 x_1$, konstantes u bitus kā $u_n \dots u_2 u_1$, kur x_1 (attiecīgi u_1) ir mazāk zīmīgais bits. Pārnesumu pēc i-tā bita apzīmēsim ar c_i, mums nepieciešams aprēķināt c_n. Tā kā pēc i-tā bita būs pārnesums, ja $u_i < x_i + c_{i-1}$, tad pārnesumus var aprēķināt ar šādām darbībām:

$$\begin{aligned} c_1 &= (\neg u_1 \& x_1) \\ c_2 &= (\neg u_2 \& x_2) \vee (c_1 \& (\neg u_2 \vee x_2)) \\ &\dots \\ c_n &= (\neg u_n \& x_n) \vee (c_{n-1} \& (\neg u_n \vee x_n)) \end{aligned}$$

Tā kā u ir konstante, tad katrā rindiņā var ievietot attiecīgo bita u_i vērtību un izteiksme vienkāršojas par:

$$c_i = \begin{cases} c_{i-1} \vee x_i, & \text{ja } u_i = 0 \\ c_{i-1} \mathbin{\&} x_i, & \text{ja } u_i = 1. \end{cases}$$

Tātad katrā solī ir ne vairāk par vienu loģisko elementu (pirmajā solī nav neviena) tad kopējā šīs loģiskās shēmas sarežģītība nepārsniedz $n-1$. Gadījumā, ja jāsalīdzina būtu pretējā virzienā ($x < u$), rezultāts būtu tāds pats. Gadījumā, ja nepieciešama stingrā vienādība, to var aizstāt ar nestingro, pievienojot gala rezultātam negāciju

$$(x \leq u) \leftrightarrow \neg(u < x).$$

Līdz ar to jebkurai salīdzināšanai nav vajadzīgs vairāk par n loģiskajiem elementiem.

Saskaitīšanu $y = x+u$ var aizstāt ar atņemšanu $y = x-u'$, kur $u' = 2^n - u$. Līdz ar to atliek noskaidrot sarežģītību atņemšanai. Kā iepriekš pierādīts, lai aprēķinātu pārnesumu c_i, katrā solī nevajag vairāk par vienu loģisko elementu. Rezultāta i-ais bits ir vieninieks, ja nepāra skaits no bitiem x_i, u_i, c_{i-1} ir vieninieki. Tādējādi atkarībā no u_i (kas ir konstante), rezultātu var aprēķināt šādi:

$$y_i = \begin{cases} (x_i \& c_{i-1}) \vee (\neg x_i \& \neg c_{i-1}), & \text{ja } u_i = 0 \\ (x_i \& \neg c_{i-1}) \vee (\neg x_i \& c_{i-1}), & \text{ja } u_i = 1. \end{cases}$$

Katrā gadījumā papildus nepieciešams ne vairāk par 5 loģiskajiem elementiem katrā solī, līdz ar to kopējā sarežģītība atņemšanai (saskaitīšanai) nav lielāka par $5n + n = 6n$.

Izvēles funkcijai $f(x, y, t)$ rezultāts ir vai nu x vai y atkarībā no bita t vērtības. Katru (i-to) izejas bitu z_i var aprēķināt pēc formulas $z_i = (x_i \& \neg t) \vee (y_i \& t)$, kopā tam nepieciešami $3n+1$ loģiskie elementi (negāciju pie t pietiek aprēķināt vienreiz, jo šis loceklis ir kopīgs visiem izejas bitiem).

Modulārajai salīdzināšanai $x = u \mod v$ loģisko shēmu konstruēsim pa soļiem. Vispārīgais algoritms sastāv no n soļiem, $t_0 = x$, un j tais solis ir šāds:

$$t_{j+1} = \begin{cases} t_j - 2^{n-j} v, & \text{ja } t_j \geq 2^{n-j} v \\ t_j, & \text{ja } t_j < 2^{n-j} v. \end{cases}$$

Katrā solī jāizveic viena salīdzināšana un viena atņemšana, pie tam atņemšana jau ietver sevī salīdzināšanu līdz ar to to kopējā sarežģītība nepārsniedz $12n$ (šīs darbības jāveic $2n$ bitu skaitļiem). Papildus izvēles bloka sarežģītība nepārsniedz $3n+1$, pie tam $\neg t$ aprēķināšana ir jau ietverta salīdzināšanas operācijas sarežģītībā, ka rezultātā šīs daļas sarežģītība samazinās līdz $3n$. Līdz ar to kopējā viena soļa sarežģītība nepārsniedz $15n$ un kopējā loģiskās shēmas sarežģītība nepārsniedz $15n^2$. ∎

Turpmāk diagrammās izvēles funkciju $f(x, y, t)$ apzīmēsim ar rombu, kurā ierakstīts nosacījums, kurš atgriež bitu t, bet malās pienākošajās bultiņās atrodamas attiecīgi x (pa kreisi) vai y (pa labi) vērtības. Daudz kur, ērtības labad, vairāku (n) argumentu disjunkcijas vai konjunkcijas arī apzīmēsim ar vienu operāciju, kaut gan tās ir $n-1$ atsevišķa operācija (disjunkcija vai konjunkcija).

2.5. Tjūringa mašīnas

Tjūringa mašīnas (TM) ir pats populārākais skaitļošanas modelis, ko izmanto teorētiskajā datorzinātnē. Čērča tēze apgalvo, ka jebko, ko var izrēķināt kādā algoritmiskā veidā, var izrēķināt arī ar Tjūringa mašīnu.

Pašām Tjūringa mašīnām arī ir vairākas modifikācijas. Pašā vienkāršākajā modelī Tjūringa mašīnai ir viena uz vienu pusi bezgalīga lente, pa lenti slīd darba galviņa, sākumā uz lentes ir uzrakstīts ieejas vārds.

Tjūringa mašīna sāk darbu stāvoklī q_0. Katrā solī lentes galviņa nolasa tekošo simbolu un, vadoties pēc stāvokļu pāreju tabulas (TM programmas), nomaina to, nomaina savu stāvokli un pabīdās vienu rūtiņu pa labi vai pa kreisi (vai paliek uz vietas). Tjūringa mašīna beidz darbu, kad tā nonāk stāvoklī q_b. Uz lentes uzrakstītais vārds tad arī ir programmas rezultāts.

Formāli par Tjūringa mašīnu sauc kortežu $(Q, \Sigma, \lambda, \Gamma, q_0, q_b, \delta)$, kur

1. Q ir tās stāvokļu telpa
2. Σ ir tās ieejas (un izejas) alfabēts
3. $\lambda \notin \Sigma$ ir tukšais simbols
4. $\Gamma = \Sigma \cup \{\lambda\}$ ir tās darba alfabēts
5. q_0 un q_b ir attiecīgi tās sākuma un beigu stāvokļi
6. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, C, R\}$ ir tās pārejas funkcija

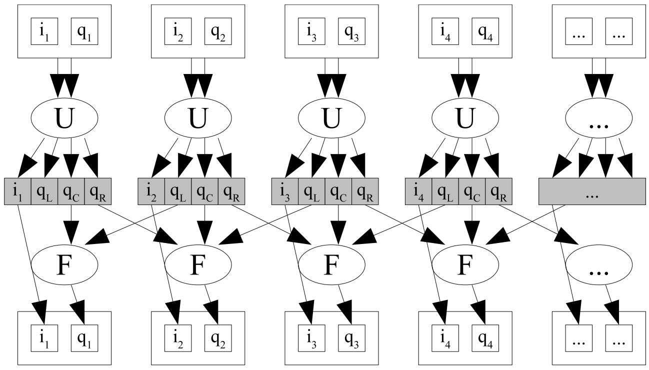
Katra Tjūringa mašīna rēķina funkciju $\Sigma^* \to \Sigma^*$, kur Σ ir tās ieejas alfabēts (mēs aplūkojam tikai tās Tjūringa mašīnas, kuras vienmēr apstājas). Īpaši var izdalīt tās Tjūringa mašīnas, kuras uz jebkuriem ieejas datiem atgriež vērtību 0 (nē) vai 1 (jā), par tām var teikt, ka tās atpazīst valodu, visu to vārdu kopu, uz kurām šī TM atgriež 1.

Nedeterminētai TM katram stāvoklim un ieejas simbolam var atbilst divas dažādas pārejas. Nedeterminēta Tjūringa mašīna pazīst vārdu x, ja tai var izvēlēties šīs pārejas tā, lai beigās uz lentes atrastos vieninieks.

Saka, ka Tjūringa mašīna darbojas laikā $f(n)$, ja jebkurai ieejai tā apstājas pēc ne vairāk kā $f(n)$ soļiem, kur n ir attiecīgo ieejas datu garums. Tjūringa mašīnas lentes sarežģītība ir $f(n)$, ja jebkurai ieejai, kuras garums ir n, tā izmanto ne vairāk par $f(n)$ lentes rūtiņām.

Ir zināms [6][31], ka Tjūringa mašīnu, kas darbojas laikā t, var simulēt ar loģisko elementu shēmu, kurā elementu skaits ir $p(t)$, kur p ir kāds polinoms, kas var būt atkarīgs no Tjūringa mašīnas stāvokļu skaita. Ir dažādas konstrukcijas, kā to var darīt, mēs lietosim to (tā nav pati optimālākā), kurā katrs Tjūringa mašīnas solis tiek simulēts ar vienu un to pašu loģisko elementu shēmu. Šī konstrukcija ir ņemta no [31].

2.6. Teorēma *Ja Tjūringa mašīna M, saņemot ieejā n bitus garus ieejas datus, rēķināšanas procesā lieto ne vairāk kā $s(n)$ lentes rūtiņas, tad katru (jebkuru) M soli var aprakstīt ar loģisko elementu shēmu, kurā ir ne vairāk kā $cs(n)$ elementu, kur c ir konstante, kas ir atkarīga no M, bet nav atkarīga no n.*



2.1. att. Tjūringa mašīnas viena soļa simulācija ar loģisko elementu shēmu

Pierādījums Vispirms pievienosim M stāvokļu kopai vienu papildus stāvokli $\tilde{q}$, ko sauksim par *tukšo stāvokli*, šis stāvoklis apzīmēs to, ka M galviņa dotajā brīdī neatrodas tekošajā rūtiņā. Iekodēsim visus M stāvokļus (ieskaitot tukšo) $m = \lceil \log(|Q|+1) \rceil$ bitos, bet visus ieejas simbolus (ieskaitot tukšo simbolu λ) — $k = \lceil \log(|\Sigma|+1) \rceil$ bitos. Katrai M lentes rūtiņai atbildīs (iekodēti) mainīgie, kas saturēs iekodētu simbolu $x \in \Sigma \cup \{\lambda\}$, kurš atrodas dotajā rūtiņā un Tjūringa mašīnas stāvokli $q \in Q \cup \{\tilde{q}\}$ pirms un pēc šī soļa. Līdz ar to dotajai shēmai būs $s(n)(m+k)$ ieejas un izejas mainīgie.

Loģiskā shēma, kas simulē vienu M soli, attēlota 2.1. zīmējumā. Tā sastāv no divu veidu elementiem — pārejas elementiem U un filtra elementiem F. Pārejas elementi apraksta lentes izmaiņas šajā solī, bet filtra elementi modelē Tjūringa mašīnas galviņas pārvietošanos.

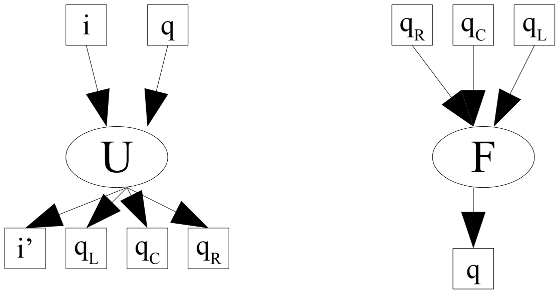
Katrs no $s(n)$ pārejas elementiem U (2.2. zīm.) lasa iekodētus ieejas simbolu i un stāvokli q (kopā $m+k$ biti), kas atbilst dotajai lentes rūtiņai, un atgriež (bināri iekodētas) vērtības i', q_L, q_C, q_R ($3m+k$ biti), kur i' ir jaunais simbols dotajā lentes rūtiņā, bet q_L, q_C, q_R ir jaunais stāvoklis šajā (q_C) un blakusesošajās (q_L, q_R) lentes rūtiņās, izejot tikai no informācijas, kas pieejama par šo rūtiņu. Ja $q = \tilde{q}$ tad šajā lentes rūtiņā galviņa neatrodas un pārejas elements nedara neko: $i' = i$ un $q_L = q_C = q_R = \tilde{q}$. Bet, ja $q \in Q$, $i = x \in \Sigma \cup \{\lambda\}$ un M pārejas tabulā ieejas pārim (x, q) atbilst pāreja $(x, q) \to (x', q', X)$, tad tas atgriež $i' = x'$ un

$$
\begin{aligned}
q_L = q', q_C = q_R = \tilde{q}, &\quad \text{ja } X = L,\\
q_C = q', q_L = q_R = \tilde{q}, &\quad \text{ja } X = C,\\
q_R = q', q_L = q_C = \tilde{q}, &\quad \text{ja } X = R.
\end{aligned}
$$

Katrs no $s(n)$ filtra elementiem F pārbauda, vai tekošajā solī galviņa nonāk uz atbilstošās rūtiņas no kreisās vai labās puses vai no tās pašas rūtiņas. Tas saņem trīs vērtības q_L, q_C, q_R no atbilstošajām rūtiņām ($3m$ ieejas biti) (Fig. 2.1) un, ja kāda no šīm vērtībām nav $\tilde{q}$, tad tas ieliek šo vērtību tekošās rūtiņas stāvokļa reģistrā, pretējā gadījumā tas tur ieraksta $\tilde{q}$.

Elementu U un F izmērs ir konstants fiksētai Tjūringa mašīnai M, katrs no šiem elementiem shēmā iekļauts $s(n)$ kopijās. Līdz ar to kopējais loģiskās shēmas izmērs nepārsniedz $(|U|+|F|)s(n) = cs(n)$. ∎

Šī loģiskā shēma apraksta jebkuru dotās Tjūringa mašīnas soli. Tātad, savienojot vairākas šādas shēmas kopā, (vienas shēmas izejas datus padodot nākamās



2.2. att. Pārejas un filtra elementi Tjūringa mašīnas viena soļa modelēšanai

shēmas ieejā) var simulēt vairāk nekā vienu dotās Tjūringa mašīnas soli.

2.7. Sekas *Ja Tjūringa mašīna M strādā laikā $T(n)$ ar lentes sarežģītību $s(n)$, tad tās darbību uz ieejas datiem garumā n var simulēt ar loģisko elementu shēmu, kurā ir ne vairāk kā $cs(n)T(n)$ elementi, kur c ir konstante, kas ir atkarīga tikai no M.*

Pierādījums Savienojot kopā $T(n)$ 2.6. teorēmā konstruētās loģiskās shēmas, iegūsim loģisko elementu shēmu, kas modelē dotās Tjūringa mašīnas darbību. Iegūtās loģiskās shēmas izmērs nepārsniedz $cs(n)T(n)$. ∎

2.8. Sekas *Ja Tjūringa mašīna M darbojas polinomiālā laikā, tad tās darbību katram ieejas datu garumam n var aprakstīt ar polinomiāla izmēra loģisko elementu shēmu.*

Pierādījums Ja $T(n) \leq p(n)$, kur p ir polinoms, tad $s(n) \leq T(n) \leq p(n)$, Tjūringa mašīna laikā $T(n)$ nevar aizpildīt vairāk par $T(n)$ rūtiņām. Līdz ar to loģiskās shēmas izmērs, kas to modelē, nepārsniedz $cs(n)T(n) \leq cp(n)^2$. ∎

Atsevišķos gadījumos Tjūringa mašīnai izdala atsevišķu darba lenti, parasti to dara tāpēc, lai varētu aplūkot lentes sarežģītības klases ar sarežģītību, kas mazāka par n.

Šādā gadījumā Tjūringa mašīnai ir divas lentes: ieejas lente, uz kuras ir rakstīts ieejas vārds, kuru var tikai lasīt, kā arī darba lente, kas sākumā ir tukšā un kuru var gan lasīt gan rakstīt. Šādai Tjūringa mašīnai par lentes sarežģītību sauc tās izmantotās darba lentes daudzumu un tas var būt arī mazāks (pat logaritmiski) par ieejas garumu.

Vēl specifiskāks modelis, brīvpieejas (random access) Tjūringa mašīna, tiks aplūkots 9. nodaļā.

2.6. Algoritmu sarežģītības klases

Algoritmu sarežģītības teorijā visbiežāk tiek aplūkoti atrisināmības uzdevumi (decision problems), tie ir uzdevumi, kur iespējamas tikai divas atbildes: jā (1) vai nē (0). Katra funkcija, kas atgriež tikai šādas divas atbildes, viennozīmīgi definē valodu: visu to ieejas vārdu kopu, uz ko tā atgriež vērtību 1.

Vienkāršākās algoritmiskās sarežģītības klases ir P, NP, PSPACE, NPSPACE, L un NL. Valoda pieder klasei P (NP), ja to var pazīt ar determinētu (nedeterminētu) TM polinomiālā laikā. Valoda pieder klasei PSPACE (NPSPACE), ja to var pazīt ar determinētu (nedeterminētu) TM ar ne vairāk kā polinomiālu lentes sarežģītību. Valoda pieder klasei L (NL), ja to var pazīt ar determinētu (nedeterminētu) TM ar ne vairāk kā logaritmisku lentes sarežģītību. Zināms, ka PSPACE = NPSPACE (Saviča teorēma).

2.9. Teorēma *Jebkurai valodai $L \in$ PSPACE var atrast tādu polinomu $s(n)$, ka eksistē Tjūringa mašīna ar bināru ieejas alfabētu, kura pazīst L ne vairāk kā $2^{s(n)}$ soļos, lietojot ne vairāk kā $s(n)$ lentes rūtiņas.*

Pierādījums Tā kā $L \in$ PSPACE, tad eksistē Tjūringa mašīna, kas darbojas binārā alfabētā un pazīst L, un kuras lentes sarežģītība ir kāds polinoms $s'(n)$. Novērtēsim tās darba laiku $T(n)$ attiecībā pret $s'(n)$.

Darba alfabēts šai TM ir $\Sigma = \{0, 1, \Lambda\}$ un tās stāvokļu kopa ir Q. Katrā solī tās konfigurācija sastāv no datiem, kas uzrakstīti uz lentes, stāvokļa un galviņas pozīcijas. Tātad kopējais iespējamais konfigurāciju skaits ir $3^{s'(n)}|Q|s'(n)$, kur $3^{s'(n)}$ ir iespējamo lentes vērtību skaits, $|Q|$ ir iespējamo stāvokļu skaits un $s'(n)$ ir iespējamo galviņas pozīciju skaits. Ja kāda no konfigurācijām rēķināšanas gaitā atkārtojas divas reizes, tad Tjūringa mašīna ieciklojas. Tā kā tai ir jāapstājas pie jebkuriem ieejas datiem, tad mēs varam secināt, ka katram Tjūringa mašīnas solim atbilst cita konfigurācija, līdz ar to soļu skaits ir ne lielāks kā $T(n) \leq 3^{s'(n)}|Q|s'(n)$.

Ņemsim $s(n) = 3|Q|s'(n)$, tas arī ir polinoms, kas acīmredzami lielāks par $s'(n)$ līdz ar to šī TM nelieto vairāk par $s(n)$ rūtiņām. Tās darbības laiku var novērtēt šādi:

$$T(n) \leq 3^{s'(n)}|Q|s'(n) < 4^{s'(n)}2^{|Q|}2^{s'(n)} = 2^{3|Q|s'(n)} = 2^{s(n)}.$$

Novērtējumā izmantota nevienādība, ka $2^x > x$ visiem naturāliem skaitļiem x. ∎

Saka, ka valodu A var polinomiālā laikā reducēt uz valodu B ($A \leq_m^P B$), ja eksistē TM, kas darbojas polinomiālā laikā un rēķina funkciju f, tādu, ka

$$x \in A \leftrightarrow f(x) \in B.$$

Valoda ir grūta kādai klasei, ja uz to var reducēt jebkuru valodu no šīs klases. Valoda ir pilnīga kādai valodu klasei, ja tā pieder šai klasei un ir tai grūta.

Polinomiālā hierarhija ir analogs daļēji rekursīvo funkciju aritmētiskajai hierarhijai ar ierobežotiem skaitļošanas resursiem. Tās pamatā ir valodu klases P un NP — valodas, ko var pazīt polinomiālā laikā ar Tjūringa mašīnu vai nedeterminētu Tjūringa mašīnu, kā arī rēķināšana ar orākulu — saka, ka funkcija pieder sarežģītības klasei A^B, ja tā pieder sarežģītības klasei A, kas papildināta ar iespēju vienā solī atrisināt kādu uzdevumu no klases B.

Polinomiālo hierarhiju formāli definē šādi: $\Sigma_0^P = P$, $\Sigma_1^P = NP$, $\Sigma_{i+1}^P = NP^{\Sigma_i^P}$. Visiem i ir spēkā, ka $\Sigma_i^P \subseteq \Sigma_{i+1}^P$, bet nav pierādīts, ka kādam i šīs divas kopas būtu dažādas, kaut gan daudzi uzskata, ka tās ir dažādas visiem i. Ar PH apzīmē visu polinomiālajā hierarhijā ietilpstošo valodu kopu:

$$PH = \bigcup_{i=0}^{+\infty} \Sigma_i^P.$$

Zināms, ka sarežģītības klase PSPACE ietver sevī visu polinomiālo hierarhiju PH, ieskaitot klases P un NP.

Sarežģītības klase P/poly satur visas valodas, kuru harakterisko funkciju jebkuram ieejas garumam n var aprakstīt ar polinomiāla (no n) izmēra loģisko elementu shēmu.

Funkcijai $F : \{0,1\}^* \to \{0,1\}$ ir polinomiāla izmēra loģisko elementu shēma ($F \in P/poly$), ja eksistē tāds polinoms $p(x)$, ka visiem n izpildās $C(F|_n) \leq p(n)$, kur ar $F|_n : \{0,1\}^n \to \{0,1\}$ apzīmē funkcijas F ierobežojumu uz ieejas virknēm garumā n.

Karpa-Liptona teorēma [23] apgalvo, ka, ja $NP \subseteq P/poly$, tad $PH = \Sigma_2^P$. Citiem vārdiem sakot, ja visām funkcijām no klases NP būtu polinomiāla izmēra loģisko elementu shēmas, tad polinomiālā hierarhija kolapsētu līdz tās otrajam līmenim Σ_2^P. Lai arī $PH = \Sigma_2^P$ ir vājāka hipotēze nekā $P = NP$, arī to ir pieņemts uzskatīt par maz ticamu.

3. nodaļa

GDA kodējumi un loģiskās shēmas reprezentācijas

Parasti GDA reprezentē tabulas formā vai arī ar stāvokļu pārejas diagrammām, kas pēc būtības ir gandrīz tas pats — diagramma savā ziņā ir tabulas vizualizācija. Tabula attēlo GDA pārejas funkciju $\delta : \Sigma \times Q \to Q$ kā tabulu, pa rindiņām uzskaitīti visi iespējamie stāvokļu un ieejas simbolu pāri, un norādīts, uz kādu nākamo stāvokli GDA pāriet, nolasot dotajā stāvoklī doto ieejas simbolu. Stāvokļu pārejas diagramma ir grafs (multigrafs), kura virsotnēs atrodas stāvokļi, kurus attēlo ar riņķiem, un no virsotnes q_1 uz virsotni q_2 ir novilkta bultiņa ar simbolu a tad un tikai tad, ja stāvoklī q_1, saņemot ieejā simbolu a, automāts pāriet stāvoklī q_2 ($\delta(a, q_1) = q_2$). Akceptējošie stāvokli tiek īpaši izcelti ar diviem koncentriskiem riņķiem.

Abas šīs reprezentācijas attēlo katru automāta stāvokli atsevišķi līdz ar to ar šīm metodēm nav iespējams efektīvi aprakstīt automātu ar lielu stāvokļu skaitu.

Automāta s stāvokļus var iekodēt $\lceil \log s \rceil$ stāvokļa bitos. Arī ieejas (un, ja nepieciešams — izejas) simbolus var iekodēt kā bitu vektorus. Katram GDA iespējami daudzi šādi kodējumi. Ja mēs neierobežojam stāvokļa bitu skaitu ar $\lceil \log s \rceil$, bet atļaujam izmantot arī vairāk bitus, tad šādu kodējumu ir bezgalīgi daudz.

Automāta pārejas funkcija šajā gadījumā saņems ieejā stāvokļa un ieejas kodējumus un atgriezīs nākamā stāvokļa vērtību (kodējumu). Šādi attēlota tā ir Būla funkcija, kuru dabiski ir aprakstīt ar loģisko elementu shēmu.

Vēl nepieciešams kaut kā aprakstīt akceptējošo stāvokļu kopu $\tilde{Q}$. Mēs to attēlosim ar loģisko elementu shēmu, kas izrēķina šīs kopas harakterisko funkciju.

Tādā veidā GDA mēs varam aprakstīt ar tā stāvokļu kopas un ieejas alfabēta kodējumiem kā arī ar divām loģisko elementu shēmām: vienu tā pārejas funkcijai un vienu akceptējošo stāvokļu kopas harakteriskajai funkcijai. Šīs shēmas mēs turpmāk sauksim attiecīgi par pārejas shēmu un akceptēšanas shēmu un kopā tās veidos GDA loģiskās shēmas reprezentāciju.

Definīcija Par kopas X *kodējumu* sauc injektīvu attēlojumu $f : \Sigma \to \{0, 1\}^{b_X}$, kas attēlo kopu X uz b_X bitu garu bitu vektoru.

Definīcija Par GDA A kodējumu $E(A)$ sauc kortežu (f_Σ, f_Q), kas sastāv no ieejas alfabēta kodējuma f_Σ un stāvokļu telpas kodējuma f_Q, pie tam $f_Q(q_0) = 0^{b_Q}$.

Definīcija Par GDA $A(Q,\Sigma,\delta,q_0,\tilde{Q})$ *pārejas shēmu* pie kodējuma $E(A) = (f_\Sigma, f_Q)$ sauc loģisko elementu shēmu F ar $b_\Sigma + b_Q$ ieejas mainīgajiem un b_Q izejas mainīgajiem, tādu, ka visiem $x \in \Sigma$ un $q \in Q$

$$q' = \delta(x,q) \Rightarrow f_Q(q') = F(f_\Sigma(x),f_Q(q)).$$

Definīcija Par GDA $A(Q,\Sigma,\delta,q_0,\tilde{Q})$ *akceptēšanas shēmu* pie kodējuma $E(A) = (f_\Sigma, f_Q)$ sauc loģisko elementu shēmu G ar b_Q ieejas mainīgajiem un vienu izejas mainīgo, ja visiem $q \in Q$ izpildās

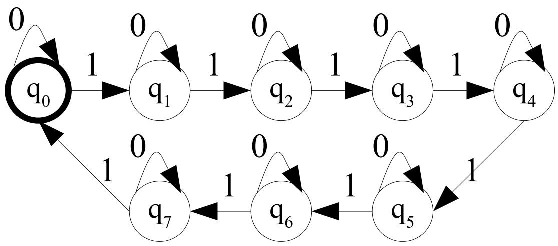
$$q \in \tilde{Q} \iff G(f_Q(q)) = 1.$$

Definīcija Par GDA $A(Q,\Sigma,\delta,q_0,\tilde{Q})$ *loģiskās shēmas reprezentāciju* pie kodējuma $E(A) = (f_\Sigma, f_Q)$ sauc loģisko elementu shēmu pāri (F, G), ja F ir tā pārejas shēma, bet G — akceptēšanas shēma pie šī kodējuma.

Citiem vārdiem sakot, pārejas shēma F saņem iekodētu ieejas simbolu $f_\Sigma(x)$ kā pirmos b_Σ ieejas mainīgos, iekodētu stāvokli $f_Q(q)$ kā nākamos b_Q ieejas mainīgos un atgriež iekodētu nākamo stāvokli $f_Q(q')$ kā tās b_Q izejas mainīgos; akceptēšanas shēma G saņem iekodētu stāvokli $f_Q(q)$ kā tās b_Q ieejas mainīgos un atgriež izejā "1" vai "0" atkarībā no tā, vai q ir akceptējošs stāvoklis vai nav.

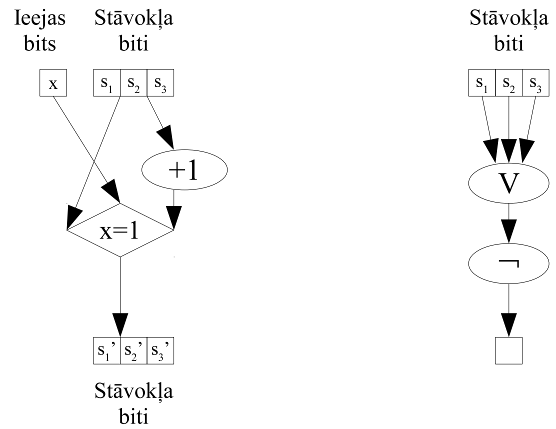
Parametru b_Σ un b_Q minimālās iespējamās vērtības ir attiecīgi $\lceil \log |\Sigma| \rceil$ un $\lceil \log |Q| \rceil$, tas seko no tā, ka funkcijas f_Σ un f_Q ir injektīvas. Bet šīs (b_Σ un b_Q) vērtības var būt arī lielākas par minimālajām un īpaši aktuāli tas ir parametram b_Q. Viens no pagaidām neatrisinātiem jautājumiem ir par to, vai neminimālu b_Q vērtību izmantošana (neoptimāla stāvokļu telpas iekodēšana) var samazināt attiecīgās loģiskās shēmas reprezentācijas izmēru.

Kā piemēru aplūkosim 3.1. zīm attēloto GDA, kurš akceptē vārdus binārā alfabētā $\Sigma = \{0, 1\}$, kuros vieninieku skaits dalās ar 8, apzīmēsim to ar A_8. Iekodēsim tā 8 stāvokļus dabīgā veidā $b_Q = 3$ stāvokļu bitos $s_1s_2s_3$ (q_i tiek kodēts kā i binārais pieraksts), bet ieejas simbolu dabīgā veidā iekodēsim vienā bitā (0 vai 1).



3.1. att. GDA A_8 stāvokļu pāreju grafs

Pie šāda kodējuma doto GDA var reprezentēt ar 3.2. zīm. attēlotajām loģiskajām shēmām. Pārejas shēma (attēlā pa kreisi) pieskaita iekodētajam stāvoklim vieninieku, ja ieejas bits ir 1, vai atstāj to nemainīgu, ja ieejas bits ir 0 (atcerēsimies, ka ar rombu apzīmē izvēles funkciju, kas atgriež savu labo ieeju, ja nosacījums, kas tajā attēlots izpildās, pretējā gadījumā tas atgriež savu kreiso ieeju). Akceptēšanas shēma (attēla pa labi) akceptē vārdu, ja GDA beidz darbu stāvoklī q_0, kas tiek kodēts ar trim nullēm.



3.2. att. GDA A_8 loģiskās shēmas reprezentācija: pārejas shēma (pa kreisi) un akceptēšanas shēma (pa labi).

Katram kodējumam un vēl jo vairāk katram GDA var atbilst vairākas loģiskās shēmas reprezentācijas. Pirmajā brīdi varētu likties, ka katra loģiskās shēmas reprezentācija reprezentē tieši viena GDA pie viena kodējuma, bet tā tas nav. Pats vienkāršākais piemērs — ņemsim kādu GDA, kas pie diviem dažādiem ieejas simboliem x un y dara vienu un to pašu, tad pamainot kodējumu: samainot vietām $f_\Sigma(x)$ un $f_\Sigma(y)$ vērtības (pārējo neaiztiekot), iegūsim jaunu kodējumu, kam derēs tā pati loģiskās shēmas reprezentācija. Šeit GDA kodējumiem atšķiras ieejas alfabēta kodējums, bet šāda situācija ir iespējama arī pie fiksēta ieejas alfabēta kodējuma (kad mainās tikai stāvokļu kodējums).

Kā vienu piemēru aplūkosim GDA A, kuram ir nesasniedzami stāvokļi, un tam ekvivalentu GDA A', kuram šie nesasniedzamie stāvokļi ir noņemti (bet sasniedzamo stāvokļu kopā šie GDA ir izomorfi). Tad jebkura A loģiskās shēmas reprezentācija ir arī A' loģiskās shēmas reprezentācija, pie kodējuma, kur A' stāvokļi kodējas tāpat, ka tiem atbilstošie A stāvokļi.

Kā citu piemēru aplūkosim gadījumu, kad kādam GDA ir ekvivalenti nesasniedzami akceptējoši stāvokļi s_1 un s_2, kuros neieiet neviena pāreja, no tiem, lasot jebkuru ieejas simbolu, automāts paliek tajā pašā stāvoklī . Tad, ņemot kādu šī GDA loģiskās shēmas reprezentāciju (F, G) pie kodējuma (f_Σ, f_Q), tā būs arī loģiskās shēmas reprezentācija pie cita kodējuma (f_Σ, f'_Q), kur $f'_Q(s_1) = f_Q(s_2)$, $f'_Q(s_2) = f_Q(s_1)$, bet visiem pārējiem stāvokļiem f_Q un f'_Q sakrīt. Tādējādi viena un tā pati loģiskās shēmas reprezentācija der viena GDA diviem dažādiem stāvokļu kodējumiem.

Bet abi šie gadījumi ir iespējami tikai tad, ja GDA ir kādi nesasniedzami stāvokļi. Gadījumā, ja tādu nav, tad pie fiksēta ieejas alfabēta kodējuma f_Σ jebkura GDA diviem dažādiem stāvokļu kodējumiem atbilst dažādas loģiskās shēmas reprezentācijas, kā arī jebkuru dažādu (neizomorfu) GDA loģiskās shēmas reprezentācijas pie jebkāda stāvokļu kodējuma ir dažādas.

3.1. Teorēma *Ja diviem neizomorfiem GDA A_1 un A_2 alfabētā Σ nav nesasniedzamu stāvokļu, tad pie fiksēta ieejas alfabēta kodējuma f_Σ jebkuras to loģiskās shēmas reprezentācijas pie jebkādiem stāvokļu kodējumiem ir dažādas.*

Pierādījums Pieņemsim, ka diviem GDA A_1 un A_2 pie kodējumiem (f^1_Σ, f^1_Q) un

(f^2_Σ, f^2_Q) atbilst viena un tā pati loģiskās shēmas reprezentācija (F, G). Ar Q_1 un Q_2 apzīmēsim to stāvokļu kopas, ar $\tilde{Q_1}$ un $\tilde{Q_2}$ — akceptējošo stāvokļu kopas un ar δ_1 un δ_2 — pārejas funkcijas. Pamatosim, ka A_1 un A_2 ir izomorfi.

Izomorfismā katram stāvoklim $q_1 \in Q_1$ atbildīs stāvoklis $q_2 \in Q_2$, tāds, ka $f^1_Q(q_1) = f^2_Q(q_2)$. Skaidrs, ka katram q_1 šāds q_2 noteikti eksistē. Pretējā gadījumā GDA A_1, nonākot stāvoklī q_1 (šeit svarīgi, ka visi stāvokļi ir sasniedzami), stāvokļu reģistrā būtu tāda vērtība, kas neatbilst nevienam A_2 stāvoklim, bet šī ir arī A_2 loģiskās shēmas reprezentācija.

Skaidrs, ka sākuma stāvokļi atbildīs viens otram, jo tie abi kodējas par bitu virkni, kas sastāv tikai no nullēm.

Ja q_1 un q_2 atbilst viens otram, $\delta_1(x, q_1) = q'_1$ un $\delta_2(x, q_2) = q'_2$, tad

$$f^1_Q(q'_1) = F(f_\Sigma(x), f^1_Q(q_1)) = F(f_\Sigma(x), f^2_Q(q_2)) = f^2_Q(q'_2),$$

tātad arī q'_1 un q'_2 atbilst viens otram.

Un, visbeidzot, akceptējošie stāvokļi abos GDA būs vieni un tie paši, jo

$$q_1 \in \tilde{Q_1} \iff G(f^1_Q(q_1)) = 1 \iff G(f^2_Q(q_2)) = 1 \iff q_2 \in \tilde{Q_2}. \quad \blacksquare$$

Tāda pat veidā viegli var pierādīt, ka viena un tā paša GDA diviem dažādiem stāvokļu kodējumiem atbilstošās loģiskās shēmas reprezentācijas būs dažādas:

3.2. Teorēma *Ja kādam GDA A_1 alfabētā Σ nav nesasniedzamu stāvokļu, tad pie fiksēta ieejas alfabēta kodējuma f_Σ tā loģiskās shēmas reprezentācijas pie dažādiem stāvokļu kodējumiem būs dažādas.*

Pierādījums Pieņemsim, ka mums diviem dažādiem stāvokļu kodējumiem f^1_Q un f^2_Q atbilst viena un tā pati loģiskās shēmas reprezentācija. Tieši tāpat kā iepriekšējajā teorēmā, mēs varam atrast šī GDA automorfismu, nodefinējot

$$q_1 \sim q_2 \iff f^1_Q(q_1) = f^2_Q(q_2).$$

Bet GDA bez nesasniedzamiem stāvokļiem vienīgais automorfisms ir identitāte, tātad $f^1_Q(q) = f^2_Q(q)$ visiem $q \in Q$, tātad abi stāvokļu kodējumi ir vienādi. $\blacksquare$

Dabīgs veids kā nokodēt stāvokļu telpu Q ir kodēt to ar $|Q|$ leksikogrāfiski pirmajām bitu virknēm garumā $\lceil \log |Q| \rceil$, šādā gadījumā mēs teiksim, ka stāvokļu kodējums ir *minimāls*. Ieejas alfabēta minimālu kodējumu definē analoģiski. GDA A kodējumu $E(A)$ sauc par minimālu, ja abi kodējumi: gan stāvokļa, gan ieejas ir minimāli.

Jebkuram minimālam kodējumam ir spēkā $b_\Sigma = \lceil \log |\Sigma| \rceil$ un $b_Q = \lceil \log |Q| \rceil$, bet pretējā virzienā šis apgalvojums ir spēkā tikai tad, ka Q un Σ ir divnieka pakāpes.

Nosacījums, ka sākuma stāvoklis tiek iekodēts par virkni, kas sastāv tikai no nullēm, 3.1. teorēmā ir būtisks, pretējā gadījumā, ja sākuma stāvokli varētu kodēt patvaļīgi, tad viena un tā pati loģiskās shēmas reprezentācija varētu derēt visiem GDA, kas iegūti no kāda viena GDA, tam nomainot sākuma stāvokli.

No otras puses šis nosacījums, ka sākuma stāvoklis attēlojas par nuļļu virkni, nav pārāk strikts. Ja mums ir kāda GDA loģiskās shēmas reprezentācija, pie stāvokļu kodējuma, kur šis nosacījums neizpildās (sākuma stāvoklis q_0 tiek kodēts par kādu bitu virkni $a_1 a_2 \ldots a_{b_Q}$, kas nesastāv tikai no nullēm), tad to var viegli pārvērst par loģiskās shēmas reprezentāciju, kur sākuma stāvoklis attēlojas par nuļļu virkni, pārdefinējot kodējumu $f'_Q(q) = f_Q(x) \oplus a_1 a_2 \dots a_{b_Q}$ un pievienojot negācijas pie tiem ieejas un izejas mainīgajiem, kuriem $a_i = 1$. Maksimums $2b_Q$ negācijas šādi jāpieliek pārejas shēmas ieejas un izejas mainīgajiem un maksimums b_Q negācijas akceptēšanas shēmas ieejas mainīgajiem, tātad kopējais abu shēmu izmēra pieaugums ir ne lielāks par $3b_Q$.

4. nodaļa

BC-sarežģītība

4.1. Definīcija un vienkāršākās īpašības

Šajā nodaļā mums beidzot viss ir sagatavots, lai varētu nodefinēt šī darba galveno jēdzienu, GDA BC-sarežģītību.

Definīcija GDA loģiskās shēmas reprezentācijas (F,G) BC-sarežģītība ir tā pārejas shēmas sarežģītības, akceptēšanas shēmas sarežģītības un stāvokļa bitu skaita summa:

$$C_{\text{BC}}((F,G)) = C(F) + C(G) + b_Q.$$

Te uzreiz jāpaskaidro, kāpēc ir vajadzīgs šis stāvokļa bitu skaits b_Q, loģiski taču būtu ņemt vienkārši abu loģisko shēmu sarežģītību summu. Šis saskaitāmais nodrošina to, lai patvaļīgi lieliem GDA nevarētu atbilst loģiskās shēmas reprezentācijas ar sarežģītību 0, kā tas būtu, piemēram, 4.5. zīmējumā redzamajam GDA.

GDA A BC-sarežģītība pie kodējuma $E(A)$ ir minimālā BC-sarežģītība, kāda ir kādai tā loģiskās shēmas reprezentācijai pie šī kodējuma, apzīmēsim to ar $C_{\text{BC}}(A, E(A))$. Attiecīgi tās A loģiskās shēmas reprezentācijas, kuru sarežģītība pie dotā kodējuma ir minimālā, sauksim par tā minimālajām loģiskās shēmas reprezentācijām pie dotā kodējuma.

Definīcija GDA A BC-sarežģītība $C_{\text{BC}}(A)$ ir minimālā sarežģītība, kāda tam ir pie kāda kodējuma:

$$C_{\text{BC}}(A) = \min\{C_{\text{BC}}(A, E(A)) : E(A) \text{ ir GDA } A \text{ kodējums}\}.$$

Attiecīgi tās A loģiskās shēmas reprezentācijas (pie kāda kodējuma), kuru sarežģītība ir vienāda ar $C_{\text{BC}}(A)$ sauksim par tā minimālajām loģiskās shēmas reprezentācijām.

Nosaukums BC-sarežģītība ir saīsinājums no "Boolean circuit", un lai arī gribētos šo sarežģītību saukt vienkāršāk, par GDA loģiskās shēmas sarežģītību, tomēr, pārejot tālāk uz regulārām valodām, izrādās, ka šis nosaukums (regulāras valodas loģiskās shēmas sarežģītība) jau ir aizņemts (skat. 10.2. nodaļu).

Definīcija Par regulāras valodas L BC-sarežģītību sauksim minimālo BC-sarežģītību, kāda ir kādam GDA, kas atpazīst šo valodu:

$$C_{\text{BC}}(L) = \min\{C_{\text{BC}}(A) : A \text{ atpazīst } L\}.$$

Kā piemēru aplūkosim iepriekšējā nodaļā jau analizēto GDA A_8, kas akceptē vārdus, kuros vieninieku skaits dalās ar 8, tā loģiskās shēmas reprezentācija redzama 3.2. zīm. Lai aprēķinātu šīs reprezentācijas BC-sarežģītību, jāaprēķina tās abu loģisko shēmu sarežģītība. Akceptēšanas shēmas sarežģītība ir 3, mums ir nepieciešama 3 mainīgo disjunkcija (2 loģiskie elementi), kā arī viena negācija. Pārejas shēmas sarežģītība sastāv no saskaitīšanas bloka "$+1$" sarežģītības, kas (3 bitiem) nepārsniedz 18 un izvēles elementa sarežģītības, kas (3 bitiem) nepārsniedz 10 (2.5. teorēma).

Līdz ar to GDA A_8 BC-sarežģītību var novērtēt kā

$$C_{\text{BC}}(A_8) \leq C(F) + C(G) + b_Q \leq 18 + 10 + 3 = 31.$$

Šis gan ir novērtējums no augšas, patiesībā var izrādīties, ka šī sarežģītība ir mazāka, jo iespējams, ka šo pārejas shēmu iespējams realizēt efektīvāk, gan arī, ka pie kāda cita kodējuma abas loģiskās shēmas ir mazākas.

Taču precīzi aprēķināt BC-sarežģītību ir grūts uzdevums, jo tas ietver sevī uzdevumu aprēķināt sarežģītību Būla funkcijām. Tāpēc visus apakšējos novērtējumus (līdzīgi kā to dara Būla funkcijām) mēs veiksim nekonstruktīvi, izmantojot Dirihlē principu. Tā mēs novērtēsim no apakšas BC-sarežģītību gan loģiskās shēmas reprezentācijām, gan GDA, gan regulārām valodām. Bet, lai to visu izveiktu gludi un vienkārši, sākumā pierādīsim vienu lemmu, uz kuras balstīsies visi tālākie novērtējumi.

4.2. BC-sarežģītības apakšējās robežas novērtēšana

BC-sarežģītība pēc savas būtības ir loģiskās shēmas sarežģītība un, lai novērtētu no apakšas BC-sarežģītību, mums ir nepieciešams novērtēt no apakšas loģiskās shēmas sarežģītību. Ir labi zināms, ka tas ir sarežģīts uzdevums un labāki tieši apakšējie novērtējumi konkrētām funkcijām par lineāriem (vispārīgajā gadījumā) nav zināmi.

Tāpēc loģisko shēmu sarežģītības novērtēšanai parasti izmanto Dirihlē principu. Ja loģisko shēmu skaits ar sarežģītību, kas nepārsniedz r, ir mazāks par pētāmās funkciju klases apjomu, tad šajā klasē būs funkcija, kuras sarežģītība ir lielāka par r.

Līdzīgu metodi izmantosim arī mēs un, tā kā tas būs vajadzīgs vairākas reizes, tad jau šeit, pašā sākumā, pierādīsim diezgan vispārīgu lemmu, kura vēlāk dažādās situācijās padarīs vienkāršu BC-sarežģītības apakšējās robežas novērtēšanu.

Fiksēsim ieejas alfabētu Σ un aplūkosim augoša apjoma GDA loģiskās shēmas reprezentāciju kopu virkni $\{\mathfrak{R}_i\}$ (šīs loģisko shēmu reprezentācijas var reprezentēt vienu un to pašu vai dažādus GDA, pie viena un tā paša vai dažādiem kodējumiem).

4.1. Lemma *Ja visiem šīm loģisko shēmu reprezentācijām atbilstošajiem kodējumiem ieejas bitu skaits ir ierobežots ($b_\Sigma < M$), tad patvaļīgām konstantēm a_1 un a_2 gandrīz visām $(F, G) \in \mathfrak{R}_i$ izpildās*

$$C_{BC}((F,G)) > \frac{\log|\mathfrak{R}_i|}{\log\log|\mathfrak{R}_i| - a_1} + a_2,$$

Pierādījums Vispirms pierādīsim, ka pie pietiekami lielām $|\mathfrak{R}_i|$ vērtībām

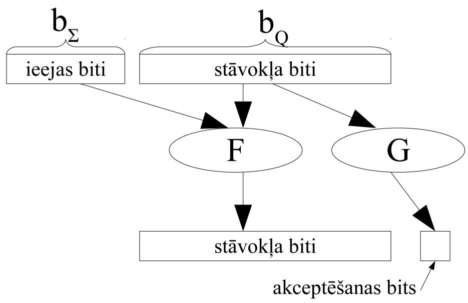
$$\frac{\log|\mathfrak{R}_i|}{\log\log|\mathfrak{R}_i| - a_1} + a_2 < \frac{\log|\mathfrak{R}_i|}{\log\log|\mathfrak{R}_i| - 2a_1}.$$

Patiešām, atstājot a_2 kreisajā pusē, bet abas daļas pārnesot uz labo pusi un vienādojot tām saucējus, iegūst nevienādību

$$\frac{a_1 \log|\mathfrak{R}_i|}{(\log\log|\mathfrak{R}_i| - a_1)(\log\log|\mathfrak{R}_i| - 2a_1)} > a_2,$$

kas ir acīmredzami pareiza pietiekami lielām $|\mathfrak{R}_i|$ vērtībām, jo kreisajā pusē esošās daļas skaitītājs aug eksponenciāli attiecībā pret $\log\log|\mathfrak{R}_i|$, bet saucējs — polinomiāli.

Aplūkosim patvaļīgu loģiskās shēmas reprezentāciju (F, G) pie kāda kodējuma ar b_Q stāvokļa bitiem un b_Σ ieejas bitiem un apvienosim loģiskās shēmas F un G vienā loģiskajā shēmā H ar $b_\Sigma + b_Q$ ieejas bitiem un $b_Q + 1$ izejas bitu tā, ka pirmie b_Q izejas biti atbilst pārejas shēmas F izejai, bet pēdējais izejas bits atbilst akceptēšanas shēmas G izejai (skat 4.1. zīm). Ievērosim, ka jebkurām divām dažādām loģiskās shēmas reprezentācijām atbilstošās shēmas H arī būs dažādas. Tāpat ievērosim, ka, iegūtās loģiskās shēmas H sarežģītība ir $C(H) = C(F) + C(G)$, tātad $C_{\mathrm{BC}}((F, G)) = C(H) + b_Q$.



4.1. att. Loģiskā shēma H.

Lai novērtētu, cik katrā kopā $|\mathfrak{R}_i|$ ir loģiskās shēmas reprezentāciju, kuru BC-sarežģītība nepārsniedz $r_i = \frac{\log|\mathfrak{R}_i|}{\log\log|\mathfrak{R}_i| - 2a_1}$, mēs varam novērtēt, cik vispār ir loģiskās shēmas reprezentāciju ar sarežģītību, kas nepārsniedz r_i. Šādu loģiskās shēmas reprezentāciju skaits noteikti nepārsniegs loģisko shēmu H skaitu, kurām $C(H) + b_Q < r_i$. Tā novērtēšanai mēs varam izmantot 2.2 teorēmu, tikai jāņem vērā, ka šīm loģiskajām shēmām H var būt dažāds ieejas un izejas mainīgo skaits.

Aplūkosim visas iespējamās b_Q vērtības ($0 \le b_Q \le r_i$), katrai no tām mums jāsaskaita loģiskās shēmas H ar $n = b_Q + b_\Sigma$ ieejas mainīgajiem, kur $0 \le b_\Sigma \le M$, $m = b_Q + 1$ izejas mainīgajiem un sarežģītību, kas nepārsniedz $r_i - b_Q$. Tāpat kā 2.2. teorēmā ar $N(n, m, c)$ apzīmēsim loģisko shēmu skaitu ar n ieejām un m izejām, kuru sarežģītība nepārsniedz c. Tad, saliekot visu kopā, iegūst, ka loģiskās shēmas reprezentāciju daļa kopā $\mathfrak{R}_i$, kuru BC-sarežģītība nepārsniedz r_i, nav lielāka par

$$\varepsilon_i = \frac{1}{|\mathfrak{R}_i|} \sum_{b_Q=0}^{r_i} \sum_{n=b_Q}^{b_Q+M} N(n, b_Q + 1, r_i - b_Q).$$

Pielietojot 2.2. teorēmu iegūstam, ka

$$\varepsilon_i < \frac{1}{|\mathfrak{R}_i|} \sum_{b_Q=0}^{r_i} \sum_{n=b_Q}^{b_Q+M} 9^{r_i - b_Q + n} (r_i - b_Q + n)^{r_i+1},$$

un, ievērojot, ka $n - b_Q \leq M$, mēs varam tālāk novērtēt, ka

$$\varepsilon_i < \frac{1}{|\mathfrak{R}_i|} \sum_{b_Q=0}^{r_i} \sum_{n=b_Q}^{b_Q+M} 9^{r_i+M} (r_i + M)^{r_i+1}.$$

Tā kā summējamā izteiksme vairs nav atkarīga no summēšanas indeksiem, tad

$$\varepsilon_i < \frac{1}{|\mathfrak{R}_i|} (r_i + 1)(M + 1) 9^{r_i+M} (r_i + M)^{r_i+1}.$$

Ja pieņem, ka $r_i > M$ (tas ir pietiekami liels), tad $r_i + M < 2r_i$, tāpēc

$$\begin{aligned} &(r_i + 1)(M + 1)9^{r_i+M}(r_i + M)^{r_i+1} < \\ &< (r_i + 1)(M + 1)9^M 9^{r_i} (2r_i)^{r_i+1} = \\ &= (r_i + 1)(M + 1)9^M (2r_i) 18^{r_i} < \\ &< (20r_i)^{r_i}, \end{aligned}$$

pēdējā nevienādība ir ekvivalenta nevienādībai

$$\left(\frac{20}{18}\right)^{r_i} > (r_i + 1)(M + 1)9^M (2r_i),$$

kas ir spēkā pietiekami lieliem r_i, jo kreisā puse aug eksponenciāli attiecībā pret r_i, bet labā — polinomiāli.

Atgriežoties pie ε_i, mūsu uzdevums ir pierādīt, ka, ja $|\mathfrak{R}_i| \to \infty$, tad $\varepsilon_i \to 0$, bet tā vietā mēs pierādīsim, ka $\log \varepsilon_i \to -\infty$:

$$\begin{aligned} \log \varepsilon_i &< \log \frac{(20r_i)^{r_i}}{|\mathfrak{R}_i|} = r_i \log(20r_i) - \log |\mathfrak{R}_i| = \\ &= \frac{\log |\mathfrak{R}_i|}{\log\log |\mathfrak{R}_i| - 2a_1} (\log 20 + \log\log |\mathfrak{R}_i| - \log(\log\log |\mathfrak{R}_i| - 2a_1)) - \log |\mathfrak{R}_i| = \\ &= \frac{\log |\mathfrak{R}_i|}{\log\log |\mathfrak{R}_i| - 2a_1} (\log 20 + 2a_1 - \log(\log\log |\mathfrak{R}_i| - 2a_1)). \end{aligned}$$

Redzams, ka, ja $|\mathfrak{R}_i| \to \infty$, tad

$$\frac{\log |\mathfrak{R}_i|}{\log\log |\mathfrak{R}_i| - 2a_1} \to \infty,$$

bet

$$(\log 20 + 2a_1 - \log(\log\log |\mathfrak{R}_i| - 2a_1)) \to -\infty,$$

no kā seko, ka $\varepsilon_i \to -\infty$. ∎

Izmantojot šo varam novērtēt arī BC-sarežģītības apakšējo robežu (gandrīz) patvaļīgai GDA kopu saimei.

4.2. Teorēma *Fiksēsim alfabētu Σ un aplūkosim augoša apjoma neizomorfu GDA bez nesasniedzamiem stāvokļiem kopu virkni $\mathfrak{A}_i$. Patvaļīgai konstantei a gandrīz visiem $A \in \mathfrak{A}_i$ izpildās*

$$C_{BC}(A) > \frac{\log |\mathfrak{A}_i|}{\log \log |\mathfrak{A}_i| - a}.$$

Pierādījums Katram $A \in \mathfrak{A}_i$ aplūkosim kādu minimālo loģiskās shēmas reprezentāciju (F, G) (kurai $C_{\text{BC}}((F, G)) = C_{\text{BC}}(A)$). Vispirms aplūkosim vienkāršotu gadījumu, kad visām šīm minimālajām loģiskās shēmas reprezentācijām (F, G) atbilst viens ieejas alfabēta kodējums f_Σ. Tā kā neizomorfiem GDA bez nesasniedzamiem stāvokļiem pie fiksēta ieejas alfabēta kodējuma arī atbilstošās loģisko shēmu reprezentācijas ir dažādas (3.1. teorēma), tad varam šo loģisko shēmu reprezentāciju kopu apzīmēt ar $\mathfrak{R}_i$. Tad, izmantojot 4.1. lemmu un to, ka $|\mathfrak{A}_i| = |\mathfrak{R}_i|$, mēs iegūsim prasīto.

Vispārīgajā gadījumā dažiem GDA no vienas kopas $\mathfrak{A}_i$ šīm atbilstošās minimālajām loģiskās shēmas reprezentācijām (F, G) var būt vienādas — bet tikai tad, ja atšķiras atbilstošie ieejas alfabēta kodējumi. Varam pieņemt, ka visiem ieejas alfabēta kodējumiem, kas tajās parādās, izpildās $b_\Sigma \le 2^{|\Sigma|}$, pretējā gadījumā kādi divi ieejas biti pieņem vienu un to pašu vērtību pie visiem $x \in \Sigma$ un visus liekos no tiem var atmest (nepalielinot BC-sarežģītību). Tādā gadījumā šādu kodējumu ir tikai ierobežots skaits, apzīmēsim to ar M. Tas nozīmē arī, ka savstarpēji vienādo loģiskās shēmas reprezentāciju skaits nepārsniedz M, tātad kopumā mums ir vismaz $|\mathfrak{A}_i|/M$ dažādas loģiskās shēmas reprezentācijas, kas atbilst katrai GDA kopai $\mathfrak{A}_i$.

Tad, pielietojot 4.1. lemmu, iegūsim, ka patvaļīgām konstantēm a_1 un a_2 gandrīz visiem $A \in \mathfrak{A}_i$ izpildās

$$C_{BC}(A) > \frac{\log \frac{|\mathfrak{A}_i|}{M}}{\log \log \frac{|\mathfrak{A}_i|}{M} - a_1} + a_2 > \frac{\log |\mathfrak{A}_i| - \log M}{\log \log |\mathfrak{A}_i| - a_1} + a_2.$$

Ņemot $a_1 = a$, $a_2 = 1$ un ievērojot, ka pietiekami lieliem i

$$\frac{\log M}{\log \log |\mathfrak{A}_i| - a} < 1,$$

iegūsim prasīto. ∎

To pašu var teikt arī par regulārām valodām.

4.3. Teorēma *Fiksēsim alfabētu Σ un aplūkosim augoša apjoma regulāru valodu kopu virkni $\mathfrak{L}_i$. Tad patvaļīgai konstantei a gandrīz visām $L \in \mathfrak{L}_i$*

$$C_{BC}(L) > \frac{\log |\mathfrak{L}_i|}{\log \log |\mathfrak{L}_i| - a}.$$

Pierādījums Katrai valodai L ņemsim GDA A ar minimālo BC-sarežģītību ($C_{\text{BC}}(A) = C_{\text{BC}}(L)$), kas šo valodu pazīst, tie visi būs dažādi un bez nesasniedzamiem stāvokļiem, tāpēc no 4.2 teorēmas seko prasītais. ∎

4.3. BC-sarežģītības atkarība no stāvokļu kodējuma

Skaidrs, ka GDA loģiskās shēmas BC-sarežģītība ir atkarīga no tā kodējuma. Sāksim ar pavisam vienkāršu novērojumu, ka jebkuram GDA var izvēlēties tādu kodējumu, lai GDA loģiskās shēmas reprezentācijas akceptēšanas shēmas sarežģītība būtu logaritmiska attiecībā pret stāvokļu skaitu.

4.4. Teorēma *Jebkuram GDA A, kura stāvokļu skaits ir s, var atrast loģiskās shēmas reprezentāciju $(F, G))$, kurai $C(G) < \lceil \log s \rceil$.*

Pierādījums Lai to panāktu, izvēlēsimies atbilstošu minimālu stāvokļu kodējumu f_Q. Ja sākuma stāvoklis q_0 ir akceptējošs, tad izvēlēsimies f_Q tā, lai tā attēlo visus akceptējošos stāvokļus par leksikogrāfiski mazākām bitu virknēm nekā noraidošos, pretējā gadījumā par leksikogrāfiski lielākām bitu virknēm nekā noraidošos stāvokļus. Pieņemsim, ka u ir leksikogrāfiski mazākā (lielākā) bitu virkne, kas atbilst kādam noraidošam stāvoklim. Tad, $q \in \tilde{Q} \iff f_Q(q) < u$ (attiecīgi $f_Q(q) > u$), kur $<$ ($>$) apzīmē bitu virkņu leksikogrāfisko salīdzināšanu. Šādai salīdzināšanai pēc 2.5. teorēmas sarežģītība nav lielāka par n, kur $n = \lceil \log s \rceil$ ir stāvokļu bitu skaits. ∎

Šī teorēma parāda, ka, pareizi izvēloties kodējumu, mēs varam būtiski samazināt akceptēšanas shēmas izmēru. Bet tas nenozīmē, ka šāda kodējuma izvēle atļaus konstruēt loģiskās shēmas reprezentāciju ar optimālu BC-sarežģītību. Var gadīties, ka, pārkārtojot stāvokļus kā 4.4. teorēmā, akceptēšanas shēmas G sarežģītība samazinās, turpretī pārejas shēmas F sarežģītība pieaug. To, ka dažreiz tas tā noteikti būs, parāda vēlāk, 4.9. teorēmā, aplūkotā valoda L_n^{Sh}. Tajā, ja stāvokļiem tiek lietots dabīgs kodējums, tad akceptēšanas shēmas G sarežģītība ir loti liela (tai jārēķina Šenona funkcija), bet pārejas shēmas F sarežģītība ir salīdzinoši neliela. No 4.4. teorēmas izriet, ka ir kāds cits (minimāls) kodējums šim pašam GDA, pie kura akceptēšanas shēmas sarežģītība ir maza (n), bet attiecīgi pārejas shēmas F sarežģītība pieaug, jo kopējā šīs valodas BC-sarežģītība ir liela.

Otrkārt, šī (4.4. teorēma) nav pati labāka optimizācija kodējuma izvēlē, ar kuras palīdzību var samazināt automāta BC-sarežģītību. Vēlāk 4.7. teorēmā tiks lietots cits kodējums, kurš optimizē nevis akceptēšanas shēmu G, bet gan pārejas shēmu, un ir asimptotiski optimāls.

Ja līdz šim mēs aplūkojām jautājumu, kā izvēlēties GDA kodējumu, lai tā BC-sarežģītība būtu pēc iespējas maza, tad mēs varam aplūkot arī pretējo, cik liela var būt BC-sarežģītība, ja mēs kodējumu izvēlamies slikti vai patvaļīgi. Jebkuram GDA vienmēr var atrast kodējumu, kura BC-sarežģītība ir liela. Šis apgalvojums vispārīgā formā ir triviāls, jo mēs vienmēr varam izvēlēties kodējumu ar patvaļīgi lielu stāvokļa bitu skaitu un tā kā visi šie stāvokļa biti ieskaitās loģiskās shēmas reprezentācijas BC-sarežģītībā, tad tā BC-sarežģītība var būt pēc patikas liela.

Tāpēc mēs varam ierobežot kodējuma izvēli un aplūkot tikai minimālos kodējumus. Viegli novērtēt (kā tas tiks izdarīts 4.7. teorēmas sākumā), ka šādā gadījumā BC-sarežģītība vairs nevar būt patvaļīgi liela, tās maksimālā vērtība ir aptuveni ks, kur k ir alfabēta izmērs, bet s — stāvokļu skaits.

Jautājums, cik tā var būt maza. Tas kaut kādā ziņā ir atkarīgs no valodas, bet visām valodām ar pietiekami lielu stāvokļu skaitu BC-sarežģītība lielākajai daļai tā minimālo kodējumu nav daudz mazāka par s. Lai to pierādītu, ņemsim patvaļīgu GDA (virkni) $\{A_s\}$, kur s ir GDA A_s stāvokļu skaits, un aplūkosim visus tā minimālos kodējumus.

4.5. Teorēma *Jebkurai GDA virknei ar augošu stāvokļu A_s gandrīz visiem A_s minimālajiem kodējumiem $E(A_s)$*

$$C_{\text{BC}}(A_s, E(A_s)) \gtrsim s.$$

Pierādījums Aplūkosim A_s minimālo kodējumu apakškopu pie kāda fiksēta ieejas simbolu (minimālā) kodējuma. Tie atšķiras tikai ar stāvokļu kodējumu, un to skaits ir $(s-1)!$, jo sākuma stāvoklis tiek kodēts par visam nullēm, bet pārējos var iekodēt patvaļīgi. Katram no šiem kodējumiem atradīsim minimālo loģiskās shēmas reprezentāciju (F, G) t.i. tādu, ka $C_{\text{BC}}((F, G)) = C_{\text{BC}}(A_s, E(A_s))$, un iegūto loģisko shēmu reprezentāciju kopu apzīmēsim ar $\mathfrak{R}_s$. No 3.2. teorēmas izriet, ka tās visas ir dažādas, tāpēc no 4.1. lemmas, ņemot $a_1 = a_2 = 0$, izriet, ka gandrīz visiem minimālajiem kodējumiem $E(A_s)$ izpildās

$$C_{\text{BC}}(A_s, E(A_s)) = C_{\text{BC}}((F, G)) > \frac{\log |\mathfrak{R}_s|}{\log \log |\mathfrak{R}_s|}.$$

Tā kā $|\mathfrak{R}_s| = (s-1)!$, tad atliek pierādīt, ka

$$\frac{\log (s-1)!}{\log \log (s-1)!} \gtrsim s.$$

Tā kā visiem $s \geq 2$ izpildās

$$\frac{s^s}{4^s} < (s-1)! < s^s,$$

tad

$$\frac{\log (s-1)!}{\log \log (s-1)!} > \frac{s \log s - 2s}{\log s + \log \log s} = s \left(1 - \frac{\log \log s + 2}{\log s + \log \log s}\right) \gtrsim s. \quad \blacksquare$$

No otras puses, šis novērtējums arī ir optimāls, jo dažiem GDA visiem minimālajiem kodējumiem BC-sarežģītība ir ar kārtu s (vai mazāka).

Aplūkosim automātus k-simbolu alfabētā, kuru stāvokļu pārejas funkcija nav atkarīga no ieejas simbola, uz visiem ieejas simboliem tie reaģē vienādi. Ņemsim patvaļīgu šādu GDA virkni ar augošu stāvokļu skaitu $\{A_s\}$, katram šādam GDA izvēlēsimies patvaļīgu minimālo kodējumu $E_s(A_s)$. Tad:

4.6. Teorēma

$$C_{\text{BC}}(A_s, E_s(A_s)) \lesssim s$$

Pierādījums Aplūkosim minimālo loģiskās shēmas reprezentāciju (F, G) dotajam GDA A_s pie dotā kodējuma $E_s(A_s)$. Tās pārejas funkcija nav atkarīga no ieejas bitiem, tāpēc tā pēc būtības ir funkcija no $\lceil \log s \rceil$ (stāvokļa) bitiem uz $\lceil \log s \rceil$ (stāvokļa) bitiem, kurai pie tam tikai pirmās s vērtības ir būtiskas, pārējās var būt patvaļīgas. Šādas loģiskās shēmas sarežģītību var novērtēt ar 2.4. teorēmas palīdzību, ņemot $\nu = s$ un $m = \lceil \log s \rceil$:

$$C(F) \lesssim \frac{s \lceil \log s \rceil}{\log s + \log \lceil \log s \rceil} \lesssim s.$$

Sarežģītību akceptēšanas shēmai G ar tās pašas 2.4. teorēmas palīdzību, ņemot $\nu = s$ un $m = 1$ var novērtēt kā

$$C(G) \lesssim \frac{s}{\log s}.$$

Līdz ar to

$$C_{\text{BC}}(A_s, E_s(A_s)) \leq C(F) + C(G) + b_Q \lesssim s + \frac{s}{\log s} + \lceil \log s \rceil \lesssim s. \quad \blacksquare$$

4.4. BC-sarežģītība salīdzinot ar stāvokļu sarežģītību

Šajā nodaļā mēs salīdzināsim GDA un regulāro valodu BC-sarežģītību ar stāvokļu sarežģītību. Vispirms parādīsim, ka šīs sarežģītības var atšķirties ne vairāk kā eksponenciāli, GDA ar s stāvokļiem BC-sarežģītība nav mazāka par $\lceil \log s \rceil$ un daudz nepārsniedz $(k-1)s$, kur k ir ieejas alfabēta izmērs.

Pēc tam aplūkosim dažus piemērus, kuros būs redzams, ka dažiem automātiem ar vienādu stāvokļu skaitu BC-sarežģītība tiešām atšķiras eksponenciāli. Nodaļas beigās ar nekonstruktīvām metodēm parādīsim, ka gandrīz visām regulārām valodām to BC-sarežģītība ir tuva maksimālajai iespējamajai vērtībai (attiecībā pret stāvokļu sarežģītību).

Bet sāksim ar BC-sarežģītības augšējās un apakšējās robežas novērtēšanu attiecībā pret stāvokļu sarežģītību.

4.7. Teorēma *Ja $|\Sigma| = k \geq 2$ tad jebkuram GDA A ar s stāvokļiem,*

$$\lceil \log s \rceil \leq C_{\text{BC}}(A) \lesssim (k-1)s.$$

Ja $|\Sigma| = 1$ tad jebkuram GDA A ar s stāvokļiem,

$$\lceil \log s \rceil \leq C_{\text{BC}}(A) \lesssim \frac{s}{\log s}.$$

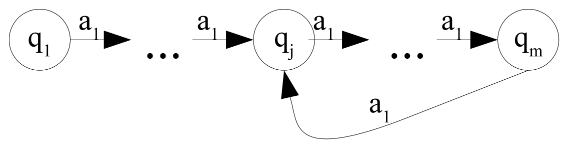
Pierādījums Apakšējā robeža. Lai iekodētu s stāvokļus, vajadzīgi vismaz $\lceil \log s \rceil$ stāvokļu biti, tāpēc jebkurai loģiskās shēmas reprezentācijai (F, G), kas reprezentē GDA ar s stāvokļiem

$$C_{\text{BC}}(F,G) = C(F) + C(G) + b_Q \geq b_Q \geq \lceil \log s \rceil \,.$$

Augšējā robeža. Ja mēs izvēlētos patvaļīgu minimālu kodējumu (ar $b_Q = \lceil \log s \rceil$ stāvokļa bitiem) un konstruētu tam optimālo loģiskās shēmas reprezentāciju (F, G), tad tās BC-sarežģītību ar 2.4. teorēmas palīdzību varētu novērtēt kā:

$$C_{\text{BC}}(F,G) = C(F) + C(G) + b_Q \lesssim \frac{ks\lceil \log s \rceil}{\log ks + \log \lceil \log s \rceil} + \frac{s}{\log s} + \lceil \log s \rceil \lesssim ks.$$

Lai uzlabotu šo rezultātu no ks līdz $(k-1)s$, izvēlēsimies tādu minimālo kodējumu, kurā stāvokļi ir sakārtoti tā, ka kādam ieejas simbolam atbilstošā pārejas funkcija ir salīdzinoši vienkārša. Īss idejas izklāsts būtu šāds: izvēlamies patvaļīgu ieejas simbolu un aplūkojam GDA stāvokļu pārejas grafu pie šī izvēlētā ieejas simbola. Tas sastāv no saistītām komponentēm, katra no kurām izskatās, kā "cilpa" ar iespējamu "asti" (4.2. zīm). Šīs komponentes var sakārtot pēc parametriem (m, j),

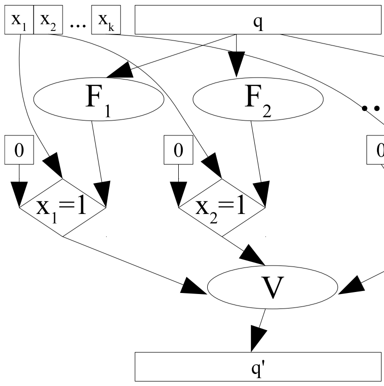


4.2. att. Viena saistītā komponente izskatās kā "cilpa" ar "asti"

un šis komponenšu sakārtojums savukārt dabīgā veidā nosaka GDA stāvokļu sakārtojumu. Stāvokļu pārejas funkcija pie šāda stāvokļu sakārtojuma pie šī ieejas simbola ir nenozīmīgi maza attiecībā pret šīs funkcijas sarežģītību pie pārējiem $(k-1)$ ieejas simboliem, tāpēc kopējā GDA BC-sarežģītība samazinās no ks uz $(k-1)s$.

Tālāk izklāstīsim šo ideju formāli. Kodēsim ieejas alfabēta $\Sigma = \{a_1, a_2 \ldots, a_k\}$ simbolus ar k bitu virknēm, tā ka a_i kodējumā visi biti būs nulles, izņemot i-o, kurš būs vieninieks. Līdz ar to, lai noskaidrotu, vai ieejas simbols ir a_i, mums pietiek pārbaudīt ieejas kodējumā i-ā bita vērtību.

Konstruēsim pārejas shēmu F katram ieejas simbolam atsevišķi, loģiskā shēma F_i rēķinās pārejas funkciju ieejas simbolam a_i (4.3. zīm). Shēmu F_1 konstruēsim īpaši (tam arī piemeklēsim stāvokļu kodējumu), bet atlikušās shēmas F_i, $i \geq 2$ konstruēsim vispārīgajā veidā, izmantojot 2.4. teorēmu.



4.3. att. Pārejas shēmas F optimāla konstruēšana

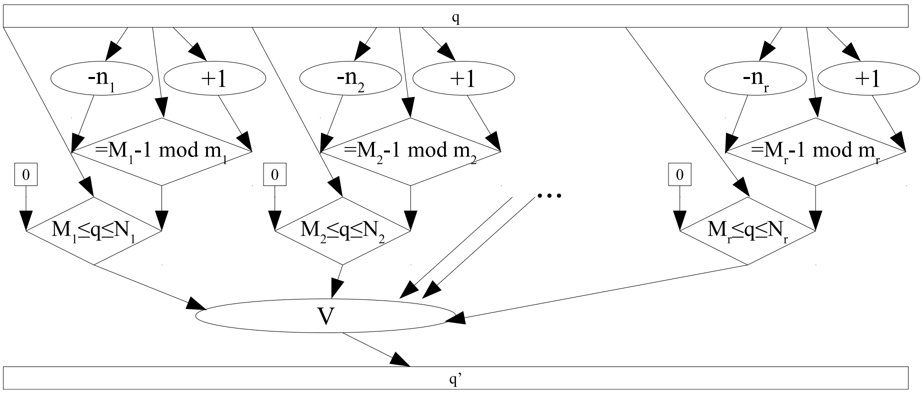
Ja mēs aplūkojam stāvokļu pārejas grafu ieejas simbolam a_1, tad tas sadalās saistītās komponentēs, katra no kurām izskatās kā cilpa ar asti (4.2. zīm).

Katra šāda komponente ir viennozīmīgi noteikta ar diviem skaitļiem m (stāvokļu skaits tajā) un j ("astes" garumu). Sakārtosim visas šīs komponentes pēc skaitļiem m un j augošā secībā, kas dabīgā veidā dos arī stāvokļu sakārtojumu (katrā komponentē iekšēji stāvokļi jau ir sakārtoti). Kodēsim katru stāvokli ar tā indeksu šajā sakārtojumā, kas mums dos minimālu stāvokļu kodējumu.

Aplūkosim visas komponentes ar parametriem (m, j), to skaitu apzīmēsim ar t un noskaidrosim, kā uz tām jādarbojas loģiskajai shēmai F_1. Ja ar M apzīmē indeksu (kodējumu) pirmajam stāvoklim (q_1) pirmajai šā tipa komponentei, tad pēdējās komponentes pēdējā stāvokļa (q_m) kodējums būs $N = M + tm - 1$. Pārejas funkcija pie šī kodējuma jebkuram šīs komponentes stāvoklim būs $q' = q+1$, izņemot katras komponentes pēdējo stāvokli (q_m), kuram tā būs $q' = q - (m-j)$. Ērtības labad apzīmēsim $n = m - j$. Tā kā visām šī tipa komponentēm ir m stāvokļi, tad stāvoklis q ir kādas (m, j) tipa komponentes pēdējais stāvoklis, ja $M \leq q \leq N$ un $q+1 = M \mod m$. Līdz ar to shēmai F_1 uz šo komponenšu tipu jārēķina funkcija:

$$\begin{cases} F_1(q) = q+1, & \text{ja } q \neq M-1 \mod m \\ F_1(q) = q-n, & \text{ja } q = M-1 \mod m. \end{cases}$$

Pieņemsim, ka mums ir r dažādu komponenšu tipu, sanumurēsim tos ar skaitļiem no 1 līdz r, attiecīgi sanumurēsim arī visus to parametrus (m_i, j_i, n_i, M_i, N_i, $1 \leq i \leq r$). Lai noteiktu, vai stāvoklis (tā kodējums) q pieder i-ajam komponenšu tipam, mums jāpārbauda, vai $M_i \le q \le N_i$. Līdz ar to loģiskajai shēmai F_1 jānoskaidro, kurā no r intervāliem $[M_i, N_i]$ dotais stāvokļa indekss atrodas, un tad jāizrēķina attiecīgajai komponentei atbilstošā pārejas funkcija (4.4. zīm.).



4.4. att. Loģiskā shēma F_1

Novērtēsim šīs loģiskās shēmas F_1 izmēru. Loģisko shēmu $(+1)$ un $(-n)$ izmēri nav lielāki par $6b_Q$, salīdzināšanai $(M \le q \le N)$ tas nepārsniedz $2b_Q$, modulārajai salīdzināšanai $q = M - 1 \mod m$ tas nepārsniedz $15b_Q^2$ un katrai no izvēles funkcijām tas ir $3b_Q + 1$ (2.5. teorēma). Disjunkcijai, kas beigās saliek kopā rezultātu no visām komponentēm, sarežģītība nepārsniedz $r \cdot b_Q$. Līdz ar to summāro loģiskās shēmas F_1 izmēru var novērtēt kā

$$C(F_1) \le r * (6b_Q + 6b_Q + 2b_Q + 15b_Q^2 + 2 * (3b_Q + 1)) + rb_Q < 38b_Q^2 \cdot r$$

(novērtējumā tika izmantots, ka $1 \le b_Q \le b_Q^2$).

Novērtēsim kāda var būt maksimāla r vērtība. Viegli ievērot, ka vislielāko dažādo komponenšu skaitu var panākt, ja tiek izmantotas visas komponentes ar pēc iespējas mazākām m vērtībām, pie tam, katra tieši vienu reizi.

Šo gadījumu arī aplūkosim, un ar u apzīmēsim maksimālo komponentes garumu (lielāko m vērtību). Katram m atbilst m dažādi komponenšu tipi ar dažādām j vērtībām ($1 \le j \le m$), līdz ar to summāri visām komponentēm garumā m ir m^2 stāvokļu. Stāvokļu skaits ir ierobežots ar s, līdz ar to visām komponentēm līdz pat garumam $m = u - 1$ kopā ir ne vairāk kā s stāvokļu:

$$\sum_{m=1}^{u-1} m^2 = \frac{(u-1)u(2u-1)}{6} < s$$

(tā kā u ir maksimālais garums, tad visas komponentes garumā u var nebūt izmantotas, līdz ar to novērtējumā summēšana ir līdz $u - 1$).

Ievērojot, ka $(u-1)(2u-1) \ge 3/4u^2$ visiem naturāliem $u \ge 2$, iegūstam, ka $u^3 < 8s$, no kurienes izriet, ka $u \le 2\sqrt[3]{s}$.

Tā kā ar garumu m mums ir ne vairāk kā m komponentes, tad maksimālo dažādo komponenšu skaitu r var novērtēt kā:

$$r \le \sum_{m=1}^{u} m = u(u+1)/2 \le u^2 \le 4\sqrt[3]{s^2}.$$

Līdz ar to loģiskās shēmas F_1 sarežģītību var novērtēt kā

$$C(F_1) \le 38b_Q^2 r \le 38(\lceil \log s \rceil)^2 \cdot 4s^{\frac{2}{3}} = 152(\lceil \log s \rceil)^2 s^{\frac{2}{3}}.$$

Visiem pārējiem ieejas simboliem atbilstošo loģisko shēmu $F_2, F_3, \ldots$ sarežģītību var novērtēt ar 2.4. teorēmas palīdzību:

$$C(F_i) \lesssim \frac{sb_Q}{\log(sb_Q)} = \frac{s\lceil \log s \rceil}{\log s + \log \lceil \log s \rceil} \lesssim s, \quad 2 \le i \le k.$$

Izvēles blokam $x_i = 1$ sarežģītība nepārsniedz $3b_Q + 1 = 3\lceil \log s \rceil + 1$, tādu ir k, tātad to kopējā sarežģītība nav lielāka par $3kb_Q + k$. Beigu disjunkcijas sarežģītība ir $(k-1)b_Q$. Tā rezultātā visas pārejas shēmas F (4.3. zīm.) sarežģītību var novērtēt kā:

$$\begin{aligned} C(F) &= C(F_1) + C(F_2) + \cdots + C(F_k) + (3kb_Q + k) + (k-1)b_Q \le \\ &\le 152(\lceil \log s \rceil)^2 s^{2/3} + (k-1)s + 4k\lceil \log s \rceil + k, \end{aligned}$$

un tā kā loceklis $152(\lceil \log s \rceil)^2 s^{2/3}$ dominē pār $4k\lceil \log s \rceil + k$ iegūstam, ka

$$C(F) \lesssim 152(\log s)^2 s^{2/3} + (k-1)s.$$

Akceptēšanas shēmas izmēru arī var novērtēt ar 2.4. teorēmas palīdzību:

$$C(G) \lesssim \frac{s}{\lceil \log s \rceil} \lesssim \frac{s}{\log s}.$$

Šis stāvokļu sakārtojums nenodrošina to, ka sākuma stāvoklis tiek kodēts ar nuļļu virkni. Lai šo izlabotu un iegūtu kādu kodējumu, kur tas tiek kodēts ar visām nullēm, kā iepriekš minēts (skat. 3. nodaļu), pietiek papildināt abas loģiskās shēmas ar ne vairāk kā $3b_Q = 3\lceil \log s \rceil$ loģiskajiem elementiem.

Saliekot to visu kopā, visas loģiskās shēmas reprezentācijas (F, G) BC-sarežģītību var novērtēt kā:

$$\begin{aligned} C_{\mathrm{BC}}((F,G)) &\le C(F) + C(G) + b_Q \lesssim \\ &\lesssim 152(\lceil \log s \rceil)^2 s^{2/3} + (k-1)s + \frac{s}{\log s} + 3\lceil \log s \rceil + \lceil \log s \rceil, \end{aligned}$$

un tā kā locekls $\frac{s}{\log s}$ dominē pār pirmo un abiem pēdējiem, tad

$$C_{\mathrm{BC}}((F,G)) \lesssim (k-1)s + \frac{s}{\log s}.$$

Divu vai vairāk ieejas simbolu gadījumā ($k \ge 2$) dominējošais saskaitāmais ir $(k-1)s$:

$$C_{\mathrm{BC}}(A) \lesssim (k-1)s + \frac{s}{\log s} \lesssim (k-1)s,$$

bet viena burta alfabētam ($k = 1$)

$$C_{\mathrm{BC}}(A) \lesssim (k-1)s + \frac{s}{\log s} = \frac{s}{\log s}. \quad \blacksquare$$

No 4.7. teorēmas tiešā veidā izriet BC-sarežģītības novērtējums regulārām valodām:

4.8. Teorēma *Ja $|\Sigma| = k$, tad jebkurai regulārai valodai L, kuras stāvokļu sarežģītība ir s*

$$\begin{aligned} \lceil \log s \rceil \leq & C_{\text{BC}}(L) \lesssim (k-1)s, && \text{ja } k \geq 2, \\ \lceil \log s \rceil \leq & C_{\text{BC}}(L) \lesssim \frac{s}{\log s}, && \text{ja } k = 1. \end{aligned}$$

Pierādījums Jebkuram GDA, kas pazīst L, ir vismaz s stāvokļi, tāpēc tā BC-sarežģītība ir lielāka par $\lceil \log s \rceil$. Ja mēs aplūkojam minimālo GDA $M(L)$, kas pazīst L, tad tā stāvokļu skaits ir s, tāpēc

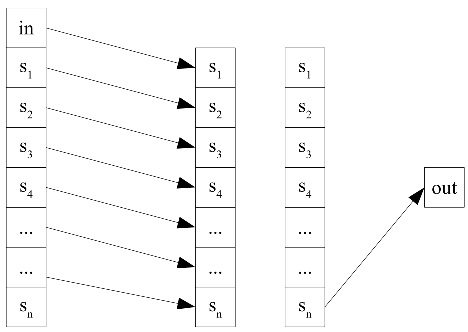
$$C_{\text{BC}}L \leq C_{\text{BC}}(M(L)) \lesssim \begin{cases} (k-1)s, & \text{ja } k \geq 2 \\ \frac{s}{\log s}, & \text{ja } k = 1. \end{cases} \quad \blacksquare$$

Tālāk aplūkosim divas regulāras valodas, kuru BC-sarežģītība ir tuva 4.8. teorēmas attiecīgi apakšējai un augšējai robežai.

Kā pirmo aplūkosim valodu L_n binārā alfabētā $\Sigma = \{0, 1\}$, tādu, ka $w \in L_n \iff w_{|w|-n+1} = 1$ (n-tais simbols no vārda beigām ir "1"). Stāvokļu sarežģītība šai valodai ir $sc(L_n) = 2^n$, jebkuram GDA, kas to pazīst, jāatceras visi pēdējie n ieejas simboli (un ar to pietiek). Bet tās BC-sarežģītība ir n.

Minimālajam GDA A_N, kas pazīst L_n, ir 2^n stāvokļu, katrs stāvoklis atbilst citai pēdējo n nolasīto (bināro) simbolu virknei. Kodēsim katru stāvokli ar šo virkni, tādā gadījumā i-ais stāvokļa bits atbildīs i-ajam (no beigām) nolasītajam ieejas simbolam. Pašus ieejas simbolus (0 un 1) dabīgā veidā kodēsim ar vienu ieejas bitu.

Loģiskas shēmas (F, G), kas reprezentē A_n šajā stāvokļu kodējumā attēlotas 4.5. zīmējumā, tām nav neviena loģiskā elementa līdz ar to A_n BC-sarežģītību nosaka tikai stāvokļa bitu skaits b_Q, kas ir tieši n. Tātad $C_{\text{BC}}(L_n) = \log sc(L_n)$, un šis piemērs parāda, ka 4.8. teorēmas apakšējā robeža ir sasniedzama.

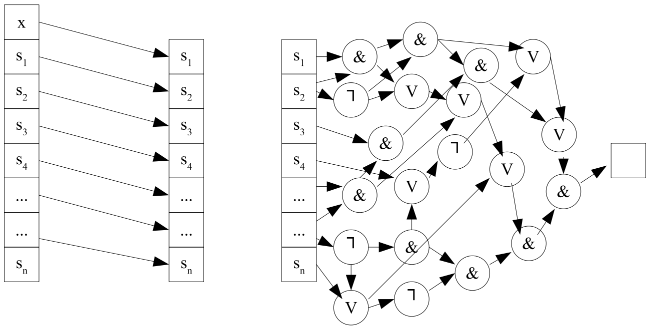


4.5. att. Pārejas shēma F (pa kreisi) un akceptēšanas shēma G (pa labi), kas reprezentē GDA A_n.

Turpināsim ar augšējo robežu. Atcerēsimies, ka ar Sh_n apzīmē Šenona funkciju no n bitiem, leksikogrāfiski pirmo Būla funkciju ar n ieejas bitiem (un vienu izejas bitu) ar maksimālo loģiskās shēmas sarežģītību. Aplūkosim valodu L_n^{Sh}, kas sastāv no tiem vārdiem $w_1 w_2 \ldots w_k$ binārā alfabētā, uz kuru pēdējiem n simboliem Šenona funkcija atgriež vieninieku:

$$w \in L_n^{Sh} \iff Sh_n(w_{k-n+1}, w_{k-n+2}, \ldots, w_k) = 1.$$

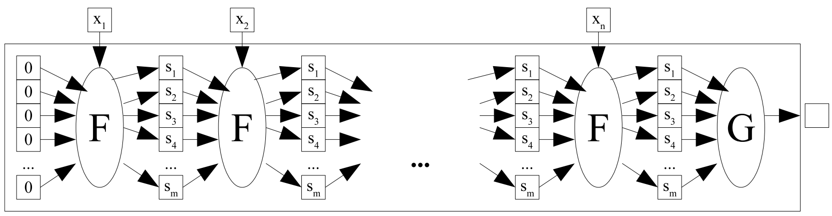
Dabisks veids, kā to pazīt, ir atcerēties GDA stāvoklī pēdējos n ieejas simbolus un, kad pienāk vārda beigas, tad akceptēt vai noraidīt šo vārdu, atkarībā no Sh_n vērtības uz šī stāvokļa. Tam būtu vajadzīgi 2^n stāvokļi, ko var kodēt n bitu virknēm tieši tāpat kā iepriekšējajā piemērā (i-ais stāvokļa bits atbilst i-ajam (no beigām) ieejas simbolam), tādā gadījumā tā loģiskās shēmas reprezentācija izskatīsies apmēram kā 4.6. zīm. Šis attēls ir shematisks un daudzajiem loģiskajiem elementiem akceptēšanas shēmā nav matemātiska pamata, būtiski ir tas, ka šajā gadījumā pārejas shēmai nav neviena loģiskā elementa (tās sarežģītība ir 0), bet akceptēšanas shēmas sarežģītība ir vismaz $2^n/n$ (2.3. teorēma). Tātad šīs loģiskās shēmas reprezentācijas BC-sarežģītība ir vismaz $2^n/n$.



4.6. att. Loģiskās shēmas reprezentācija GDA, kas pazīst valodu L_n^{Sh}: pārejas shēma (pa kreisi) un akceptēšanas shēma (pa labi).

Taču nav teikts, ka šī loģiskās shēmas reprezentācija ir optimāla, varētu taču gadīties, ka pie cita stāvokļu kodējuma vai izvēloties kādu citu (ekvivalentu) GDA, akceptēšanas shēmas sarežģītība samazinās. Nākamā teorēma pierāda, ka būtiski šo loģiskās shēmas reprezentāciju uzlabot nevar, jo šīs valodas BC-sarežģītība ir vismaz $2^n/n^2$.

4.9. Teorēma *Valodas L_n^{Sh} BC-sarežģītība ir vismaz $2^n/n^2$.*



4.7. att. Loģiskās shēmas konstruēšana Šenona funkcijai Sh_n no GDA A_n loģiskās shēmas reprezentācijas (F, G).

Pierādījums Pieņemsim, ka loģisko shēmu pāris (F, G) reprezentē kādu GDA A_n, kas atpazīst L_n^{Sh}. Varam pieņemt, ka F ir viens ieejas bits, kurā tiek padots ieejas simbols un m ieejas biti, kuros tiek padots iekodēts stāvoklis. (m var atšķirties dažādām loģiskās shēmas reprezentācijām). Savienojot n loģiskās shēmas F ar vienu shēmu G kā tas parādīts 4.7. zīmējumā (stāvokļa izeja no j-tās shēmas F tiek padota $j+1$-ajai shēmai F kā ieeja), iegūst loģisko shēmu, kura modelē A_n darbību uz vārda garumā n, tātad šī loģiskā shēma rēķina Šenona funkciju Sh_n.

Tās sarežģītība ir $nC(F) + C(G)$, bet no 2.3. teorēmas izriet, ka šīs loģiskās shēmas sarežģītība ir vismaz $2^n/n$. Tātad $nC(F) + C(G) > 2^n/n$, no kurienes mēs iegūstam, ka

$$C_{\mathrm{BC}}(F, G) = C(F) + C(G) + m > \frac{nC(F) + C(G)}{n} > 2^n/n^2. \quad \blacksquare$$

Stāvokļu sarežģītība valodai L_n^{Sh} nav lielāka par 2^n, pietiek atcerēties pēdējos n ieejas simbolus. Apzīmējot $sc(L_n^{Sh}) = s$ un zinot, ka $s < 2^n$, iegūstam, ka

$$C_{\mathrm{BC}}(L_n^{Sh}) \geq \frac{2^n}{n^2} \geq \frac{s}{(\log s)^2}.$$

Var ievērot, ka L_n^{Sh} BC-sarežģītība atrodas diezgan tuvu 4.8. teorēmas augšējai robežai, lai arī to nesasniedz (jebkurai valodai L binārā alfabētā, kuras stāvokļu sarežģītība ir s, $C_{\mathrm{BC}}(L) \lesssim s$). Būtu interesanti uzkonstruēt kādu valodu, kas šo augšējo robežu sasniedz, bet tas nav tik vienkārši.

Viens veids, kā to varētu mēģināt izdarīt, būtu ņemt par pamatu kādu sarežģītu $(n \to n)$ bitu funkciju, kuras loģiskās shēmas sarežģītība ir ar kārtu 2^n, un likt to par GDA pārejas funkciju. Bet tad nav skaidrs, kā pamatot, ka pie cita stāvokļu kodējuma šī pārejas funkcija nekļūst vienkāršāka.

Tāpēc, lai pamatotu, ka 4.8. teorēmas augšējā robeža ir sasniedzama, izmantosim nekonstruktīvas metodes (kaut gan arī Šenona funkcijas izmantošanu iepriekšējā piemērā nevar uzskatīt par īpaši konstruktīvu). Patiesībā izrādās, ka 4.8 teorēmas augšējo robežu sasniedz gandrīz visas regulāras valodas.

Atcerēsimies, ka ar $\mathfrak{L}_s^k$ apzīmē regulāro valodu kopu k-simbolu alfabētā, kuru stāvokļu sarežģītība nepārsniedz s.

4.10. Teorēma *Gandrīz visām valodām* $L \in \mathfrak{L}_s^k$

$$\begin{aligned} C_{\mathrm{BC}}(L) &\gtrsim (k-1)s, && \text{ja } k \geq 2,\\ C_{\mathrm{BC}}(L) &> \frac{s}{\log s}, && \text{ja } k = 1. \end{aligned}$$

Pierādījums Ja mēs $\mathfrak{L}_s^k$ pielietojam 4.3. teorēmu ar $a = 0$, tad mums atliek pierādīt, ka

$$\begin{aligned} \frac{\log |\mathfrak{L}_s^k|}{\log\log |\mathfrak{L}_s^k|} &\gtrsim (k-1)s, && \text{ja } k \geq 2, \text{ un}\\ \frac{\log |\mathfrak{L}_s^1|}{\log\log |\mathfrak{L}_s^1|} &\geq \frac{s}{\log s}, && \text{ja } k = 1. \end{aligned}$$

No 2.1. teorēmas izriet, ka $|\mathfrak{L}_s^k| \geq s^{(k-1)s}$, ja $k \geq 2$, un $|\mathfrak{L}_s^1| \geq 2^s$. Tāpēc, ja $k \geq 2$, tad

$$\frac{\log |\mathfrak{L}_s^k|}{\log\log |\mathfrak{L}_s^k|} \geq \frac{(k-1)s \log s}{\log s + \log\log s + \log(k-1)} \gtrsim (k-1)s,$$

bet, ja $k = 1$, tad

$$\frac{\log |\mathfrak{L}_s^1|}{\log\log |\mathfrak{L}_s^1|} \geq \frac{\log 2^s}{\log\log 2^s} = \frac{s}{\log s}. \quad \blacksquare$$

4.5. Šenona efekts BC-sarežģītībai

Aplūkojot vienlaicīgi 4.8. un 4.10. teorēmas, viegli ievērot, ka augšējie novērtējumi pirmajā sakrīt ar apakšējiem novērtējumiem otrajā no tām. Tas nozīmē, ka regulāru valodu BC-sarežģītībai ir spēkā Šenona efekts: gandrīz visām regulārām valodām to BC-sarežģītība ir tuvu savai maksimālajai iespējamajai vērtībai (attiecībā pret stāvokļu sarežģītību).

4.11. Sekas *Gandrīz visām valodām* $L \in \mathfrak{L}_s^k$

$$
\begin{aligned}
(k-1)s \lesssim C_{\mathrm{BC}}(L) &\lesssim (k-1)s, && \text{ja } k \geq 2,\\
\frac{s}{\log s} < C_{\mathrm{BC}}(L) &\lesssim \frac{s}{\log s}, && \text{ja } k = 1.
\end{aligned}
$$

5. nodaļa

BC-sarežģītība un nedeterminētā stāvokļu sarežģītība

Šajā nodaļā salīdzināsim regulāru valodu BC-sarežģītību ar to nedeterminēto stāvokļu sarežģītību. Pats vienkāršākais kā to varētu izdarīt ir caur (determinēto) stāvokļu sarežģītību. Ir vispārzināms, ka jebkurai regulārai valodai L, kurai $nsc(L) = s$ (nedeterminētā stāvokļu sarežģītība ir s)

$$s \leq sc(L) \leq 2^s.$$

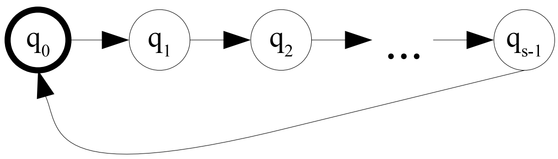
Saliekot šo kopā ar 4.8. teorēmu, kurā pamatots, ka pie $|\Sigma| = k \geq 2$ jebkurai regulārai valodai L

$$\lceil \log (sc(L)) \rceil \leq C_{\text{BC}}(L) \lesssim (k-1)sc(L),$$

varam iegūt pirmo novērtējumu valodas L BC-sarežģītībai attiecībā pret tās nedeterminēto stāvokļu sarežģītību s:

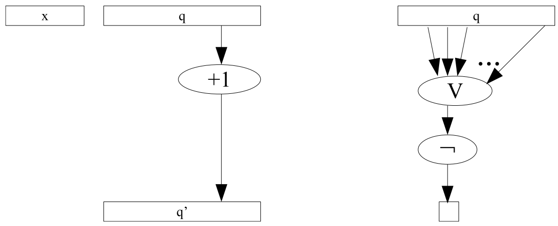
$$\lceil \log s \rceil \leq C_{\text{BC}}(L) \lesssim (k-1)2^s. \tag{5.1}$$

Apakšējo robežu šim novērtējumam īpaši uzlabot nemaz nevar. Lai to pamanītu, pietiek atcerēties, ka ir valodas, kuru nedeterminētā stāvokļa sarežģītība sakrīt ar stāvokļu sarežģītību. Kā vienu šādu piemēru aplūkosim valodu L_s, kas akceptē visus vārdus, kuru garums dalās ar s (neatkarīgi no ieejas simboliem). Šai valodai $nsc(L) = sc(L) = s$, tās minimālais GDA sakrīt ar GNA un ir attēlots 5.1. zīmējumā.



5.1. att. Minimālais GDA (un GNA), kas pazīst valodu L_s, katra pāreja atbilst jebkuram ieejas simbolam

Novērtēsim šīs valodas BC-sarežģītību un vienkāršības pēc aplūkosim gadījumu, kad s ir divnieka pakāpe. Kodēsim tās stāvokļus dabīgā veidā ar bitu virknēm no 0 līdz $s-1$, iegūstot minimālu stāvokļu kodējumu ar $b_Q = \log s$ stāvokļu bitiem, ieejas alfabētu Σ varam kodēt patvaļīgi, tas BC-sarežģītību neietekmēs, jo šī GDA pārejas funkcija nav atkarīga no ieejas simbola. Loģiskās shēmas reprezentācija šai valodai ir parādīta 5.2. zīmējumā. Pārejas shēma stāvokļu reģistram katrā solī pieskaita vieninieku, akceptēšanas shēma akceptē ieeju, ja stāvoklis ir 0.



5.2. att. Loģiskās shēmas reprezentācija minimālajam GDA, kas pazīst L_s, pārejas shēma F (pa kreisi) un akceptēšanas shēma G (pa labi).

Loģiskās shēmas F sarežģītība ir $6b_Q$ (tā sastāv tikai no saskaitīšanas), akceptēšanas shēmas sarežģītība ir b_Q ($b_Q - 1$ disjunkcija un viena negācija), līdz ar to

$$C_{\text{BC}}(L_s) \leq C(F) + C(G) + b_Q \leq 6b_Q + b_Q + b_Q = 8b_Q = 8 \log s.$$

Redzams, ka šis novērtējums atšķiras no 5.1. novērtējuma apakšējās robežas tikai konstantu skaitu (8) reižu.

Turpretī augšējo novērtējumu 5.1. formulā var uzlabot eksponenciāli. Mūsu novērtējumā zemāk būs redzams, ka determinizācijas procesā iegūtā GDA BC-sarežģītība ir ne vairāk kā kvadrātiski lielāka par sākotnējā GNA stāvokļu skaitu. No tā var secināt, ka tajos gadījumos, kad stāvokļu skaits determinizācijas procesā tiešām pieaug eksponenciāli, iegūtajam automātam ir vienkārša struktūra un tā BC-sarežģītība ir relatīvi neliela.

Mēs sāksim ar pavisam vienkāršu BC-sarežģītības novērtējumu GDA, kas iegūts, determinizējot GNA, un turpināsim ar aizvien precīzākiem novērtējumiem, mēģinot pietuvoties Šenona efektam arī nedeterminētās stāvokļu sarežģītības gadījumā.

5.1. Teorēma *Ja regulāru valodu L k-simbolu alfabētā Σ var pazīt ar s-stāvokļu GNA, kuram ir t pārejas, tad $C_{\text{BC}}(L) \leq t + (k+1)s$.*

Pierādījums Pieņemsim, ka valodu L pazīst GNA N, kura stāvokļu kopa ir $Q_N = \{q_1, q_2, \ldots, q_s\}$. Aplūkosim GDA A, kurš ir iegūts, determinizējot GNA N, un konstruēsim tam atbilstošu loģiskās shēmas reprezentāciju (un kodējumu), kura BC-sarežģītība nepārsniegs $t + (k+1)s$. GDA A stāvokļu kopa ir 2^{Q_N} (Q_N visu apakškopu kopa), tajā ir 2^s stāvokļu (no kuriem daži var nebūt sasniedzami), kurus var iekodēt s stāvokļa bitos. Patvaļīgu apakškopu $S \subseteq Q_N$ mēs varam kodēt ar bitu virkni $f_Q(S) = z_1, \ldots, z_s$, tādu, ka

$$z_i = 1 \iff q_i \in S.$$

Katrs stāvokļa bits z_i šajā kodējumā atbildīs vienam GNA N stāvoklim (q_i). Ieejas alfabētu $\Sigma = \{a_1, \ldots, a_k\}$ kodēsim ar k bitiem $f_\Sigma(a_m) = x_1, \ldots, x_k$, tā, ka katram a_m atbildīs bitu virkne, kam visur ir nulles, izņemot m-to pozīciju, kur ir vieninieks.

Tālāk mums jākonstruē pārejas shēma un akceptēšanas shēma. Pārejas shēma F saņems ieejā nokodētu ieejas simbolu un stāvokli $x_1 \dots x_k z_1 \dots z_s$ un izejā atgriezīs nokodētu stāvokli $z'_1 \dots z'_s$. Akceptēšanas shēma saņems ieejā nokodētu stāvokli $z_1 \dots z_s$ un atgriezīs vienu bitu: vieninieku, tad ja šis stāvoklis ir akceptējošs, un nulli, ja nav.

GDA A pārejas shēmu F konstruēsim no GNA N pārejas funkcijas $\delta : \Sigma \times Q_N \to 2^{Q_N}$. Pieņemsim, ka GNA atrodas savā stāvokļu apakškopā S, bet pēc simbola $x \in \Sigma$ nolasīšanas, tas atrodas savā stāvokļu apakškopā S'. Apzīmēsim ar $Q^i_m \subseteq Q$ to GNA N stāvokļu apakškopu, no kuras, nolasot simbolu $a_m \in \Sigma$, var pāriet uz stāvokli q_i:

$$q \in Q^i_m \iff q_i \in \delta(q, a_m).$$

Izmantojot šo apzīmējumu, varam uzrakstīt nosacījumu, kādā gadījumā $q_i \in S'$:

$$q_i \in S' \iff (S \cap Q^i_m) \neq \emptyset.$$

Tas nozīmē, ka Būla funkcija, kas jārēķina i-ajā izejas bitā, ir:

$$z'_i = \bigvee_{m=1}^{k} \left(x_m \& \bigvee_{q_j \in Q^i_m} z_j \right). \tag{5.2}$$

Lai par to pārliecinātos, var ievērot, ka ieejas bita x_m vērtība ir 1 tad un tikai tad, ja tiek ielasīts simbols a_m. Tātad tikai viens no m disjunkcijas locekļiem var atgriezt vieninieku, un tas to dara tad, kad iekšēja bloka vērtība $\bigvee_{q_j \in Q^i_m} z_j$ ir vieninieks — tas ir tad un tikai tad, ja $S \cap Q^i_m \neq \emptyset$.

Loģiskā shēma, kas katrā izejas bitā z'_i rēķina (5.2) formulu, arī būs pārejas shēma F šajā reprezentācijā. Aprēķināsim tās izmēru. Katrā no iekšējiem blokiem $x_m \& \bigvee_{q_j \in Q^i_m} z_j$ ir tieši $|Q^i_m| - 1$ disjunkcija un viena konjunkcija, kopā tieši $|Q^i_m|$ loģiskie elementi (gadījumā, ja $Q^i_m = \emptyset$, nav vajadzīga arī konjunkcija, tas nozīmē, ka arī šajā situācijā loģisko elementu skaits ir tieši $|Q^i_m| = 0$). Tātad kopējais šo iekšējo bloku izmērs ir

$$\sum_{i=1}^{s} \sum_{m=1}^{k} |Q^i_m| = t.$$

Katram izejas simbolam z'_i ir viena ārējā disjunkcija $\bigvee_{m=1}^{k}$ (kas sastāv no $k - 1$ disjunkcijas), kopā tie ir $s(k-1)$ loģiskie elementi. Tātad pārejas shēmas F loģisko elementu skaits ir $t + (k - 1)s$.

Akceptēšanas shēma G ir ļoti vienkārša — tā ir disjunkcija no visiem akceptējošajiem stāvokļiem atbilstošajiem bitiem z_i. Tātad tās izmērs nepārsniedz s.

Saliekot to visu kopā, iegūstam, ka šīs loģiskās shēmas reprezentācijas BC-sarežģītība ir

$$C_{\mathrm{BC}}(F, G) = C(F) + C(G) + b_Q \leq t + (k - 1)s + s + s = t + (k + 1)s. \qquad \blacksquare$$

Tā kā GNA pāreju skaits nepārsniedz ks^2, tad

5.2. Sekas *Visām valodām $L \in \mathfrak{N}^k_s$:*

$$C_{\mathrm{BC}}(L) \leq ks^2 + (k + 1)s.$$

Novērtējums 5.1 ir labs, ja GNA pāreju skaits nav liels (kā piemēram gadījumā, ja GNA ir "pretējā virzienā pagriezts" GDA), bet vispārīgajā gadījumā, kad pāreju skaits tuvojas tā vidējai vērtībai $kn^2/2$, to var uzlabot. Uzlabojuma galvenā ideja ir atkārtoti izmantot jau esošās disjunkcijas bloku $\bigvee_{q_j \in Q^i_m} z_j$ būvēšanā.

Sekojošā teorēma ir analogs 4.8. teorēmai nedeterminētās stāvokļu sarežģītības gadījumā, tās augšējā robeža uzlabo 5.2. teorēmas novērtējumu par kārtu $\log s$.

5.3. Teorēma *Visām valodām $L \in \mathfrak{N}^k_s$:*

$$\lceil \log s \rceil \leq C_{\mathrm{BC}}(L) \lesssim \frac{ks^2}{\log s}.$$

Pierādījums Apakšējā robeža seko no tā, ka$C_{\mathrm{BC}} L \leq \lceil \log (sc(L) \rceil \leq \lceil \log (nsc(L) \rceil)$.

Augšējai robežai izmantosim to pašu konstrukciju, ko 5.1. teorēmā, tikai optimālāk tiks veikta bloku $\bigvee_{q \in Q^i_m} q$ konstrukcija pārejas shēmā F. Optimizācija tiks balstīta uz to, ka vienā blokā iekļautās disjunkcijas daļēji var tikt izmantotas citos blokos.

Kopā mums nepieciešams izveidot ks šādas disjunkcijas atbilstoši ks kopām Q^i_m, katra disjunkcijā ir iekļauta daļa no stāvokļa bitiem z_i. Sadalīsim visus stāvokļa bitus grupās pa c (konstanti c izvēlēsimies vēlāk), kopumā mums būs $\lceil \frac{s}{c} \rceil$ šādas grupas.

Katras grupas ietvaros izveidosim visas $2^c - 1$ iespējamās mainīgo disjunkcijas, tam nepieciešami tieši $2^c - c - 1$ loģiskie elementi. Mums jau ir visas disjunkcijas, kas sastāv no viena elementa (ieejas biti), no tiem var viegli izveidot visas disjunkcijas, kas satur tieši divus ieejas bitus, utt., katrā solī, no disjunkcijām, kas sastāv no i mainīgajiem un pašiem ieejas mainīgajiem var izveidot visas disjunkcijas, kas sastāv tieši no $i + 1$ mainīgā, katras šādas disjunkcijas izveidei nepieciešams tieši viens $\vee$ elements. Tātad kopā mums nepieciešami tieši tik loģisko elementu, cik mums ir izejas mainīgo mīnus tik, cik mums jau ir dots. Tā kā mums nepieciešama $2^c - 1$ vērtība un c jau mums ir dotas, tad kopā mums nepieciešami $2^c - c - 1$ loģiskie elementi. Visām grupām kopā tie būs $\lceil \frac{s}{c} \rceil (2^c - c - 1)$ loģiskie elementi.

Katru no ks izejām veidosim, ņemot disjunkciju no atbilstošajām jau izrēķinātajām grupu vērtībām. Tā kā ir $\lceil \frac{s}{c} \rceil$ grupas un vienai izejai nepieciešama $\lceil \frac{s}{c} \rceil - 1$ disjunkcija, tad visām ks izejām nepieciešamas $ks(\lceil \frac{s}{c} \rceil - 1)$ disjunkcijas. Tātad kopējas elementu skaits, kas vajadzīgs visu bloku $\bigvee_{q \in Q^i_m} q$ izveidei ir

$$ks \left(\left\lceil \frac{s}{c} \right\rceil - 1 \right) + \left\lceil \frac{s}{c} \right\rceil (2^c - c - 1)$$

Tieši tāpat ka 5.1 teorēmā, vajadzīgas vēl ir $(k-1)s$ ārējās disjunkcijas $\bigvee^k_{m=1}$, jāpierēķina vēl arī ks konjunkcijas, kuras iepriekšējā teorēmā tika pieskaitītas klāt pie disjunkcijām. Tas būtu viss pārejas shēmai F. BC-sarežģītībai vēl nāk klāt akceptēšanas shēmas G sarežģītība (ne lielāka par s) un stāvokļa bitu skaits s. Tātad BC-sarežģītību šai reprezentācijai mēs varam novērtēt kā

$$C_{\mathrm{BC}}(A) \leq ks \left(\left\lceil \frac{s}{c} \right\rceil - 1 \right) + \left\lceil \frac{s}{c} \right\rceil (2^c - c - 1) + (k-1)s + ks + s + s$$

un, savelkot līdzīgos locekļus un novērtējot $\lceil \frac{s}{c} \rceil (-c - 1) + s \leq 0$, mēs iegūstam

$$C_{\mathrm{BC}}(A) \leq \left\lceil \frac{s}{c} \right\rceil (ks + 2^c) + ks.$$

Atliek noteikt konstantes c vērtību. Matemātiskās analīzes metodes šeit īsti nederēs, bet tā kā mūsu novērtējumi ir asimptotiski, tad derēs arī empīriski piemeklēta vērtība $c = \lceil \log s - \log\log s \rceil$. Tad $\lceil \frac{s}{c} \rceil \leq \frac{s}{\log s - \log\log s - 1}$, $2^c \leq 2^{\log s - \log\log s + 1} = \frac{2s}{\log s}$ un

$$C_{\text{BC}}(A) \leq \left(\frac{s}{\log s - \log\log s - 1} \right) \left(ks + \frac{2s}{\log s} \right) + ks \lesssim \frac{ks^2}{\log s} + \frac{2s^2}{(\log s)^2} + ks,$$

un ievērojot, ka loceklis $\frac{ks^2}{\log s}$ dominē pār pārējiem, secinām, ka

$$C_{\text{BC}}(A) \lesssim \frac{ks^2}{\log s}.$$

Ar to arī pierādījumu varētu beigt, bet prasās vēl neliels neformāls pamatojums, kāpēc šāda c vērtība ir optimāla un vai ar kādu citu c vērtību nevar iegūt labāku novērtējumu. Vienkāršības pēc atmetīsim visas noapaļošanas un aplūkosim izteiksmi

$$C_{\text{BC}}(A) \leq \left\lceil \frac{s}{c} \right\rceil (ks + 2^c) + ks \approx \frac{ks^2}{c} + \frac{s2^c}{c}.$$

Ja mēs ņemtu $c \geq \log ks$, tad jau viens pats otrais saskaitāmais $\frac{s2^c}{c} \geq \frac{ks^2}{\log s + \log k} \gtrsim \frac{ks^2}{\log s}$ sasniegtu asimptotisko novērtējumu, bet, ja $c \leq \log ks$, tad to savukārt sasniedz viens pats pirmais: $\frac{ks^2}{c} \gtrsim \frac{ks^2}{\log s}$. Tātad, ne uz vienu, ne uz otru pusi mainot c, mēs šaja konstrukcijā neko asimptotiski labāku iegūt nevaram. ∎

Kā jau tas tika atzīmēts nodaļas sākumā, 5.3. teorēmas apakšējā robeža ir tuvu sasniedzama. Lai parādītu, ka arī šī augšējā robeža ir tuvu sasniedzama, rīkosimies tāpat kā stāvokļu sarežģītības gadījumā un novērtēsim BC-sarežģītību gandrīz visām valodām.

5.4. Teorēma *Gandrīz visām valodām $L \in \mathfrak{N}_s^k$*

$$C_{\text{BC}}(L) > \frac{(k-1)s^2}{2\log s}, \textit{ja } k \geq 2,$$
$$C_{\text{BC}}(L) > \frac{s}{\log s}, \textit{ja } k = 1.$$

Pierādījums Gadījumā, kad $k \geq 2$, no 4.3. teorēmas prasītais būtu spēkā, ja izdotos pierādīt, ka kādai konstantei a

$$\frac{\log |\mathfrak{N}_s^k|}{\log\log |\mathfrak{N}_s^k| - a} \geq \frac{(k-1)s^2}{2\log s}.$$

No 2.1. teorēmas zināms, ka $|\mathfrak{N}_s^k| \geq 2^{(k-1)s^2}$, tāpēc

$$\frac{\log |\mathfrak{N}_s^k|}{\log\log |\mathfrak{N}_s^k| - a} \geq \frac{\log 2^{(k-1)s^2}}{\log\log 2^{(k-1)s^2} - a} = \frac{(k-1)s^2}{2\log s + \log(k-1) - a} = \frac{(k-1)s^2}{2\log s},$$

lai iegūtu pēdējo vienādību, izvēlēsimies $a = \log(k-1)$.

Savukārt, lai pierādītu teorēmas apgalvojumu, gadījumam, kad $k = 1$, izmantosim 4.3. teorēmu ar konstanti $a = 0$. Šādā gadījumā mums atliek pierādīt, ka

$$\frac{\log |\mathfrak{N}_s^1|}{\log\log |\mathfrak{N}_s^1|} \geq \frac{s}{\log s}.$$

Tā kā $|\mathfrak{N}_s^1| \geq 2^s$, (2.1. teorēma), tad

$$\frac{\log|\mathfrak{N}_s^1|}{\log\log|\mathfrak{N}_s^1|} \geq \frac{\log 2^s}{\log\log 2^s} = \frac{s}{\log s}. \quad \blacksquare$$

Saliekot kopā augšējos un apakšējos novērtējumus (5.3. un 5.4. teorēmas), mēs iegūstam sekojošu ainu:

5.5. Sekas *Gandrīz visām valodām $L \in \mathfrak{N}_s^k$*

$$\frac{(k-1)s^2}{2\log s} < C_{\text{BC}}(L) \lesssim \frac{ks^2}{\log s}, \qquad \textit{ja } k \geq 2,$$

$$\frac{s}{\log s} < C_{\text{BC}}(L) \lesssim \frac{s^2}{\log s}, \qquad \textit{ja } k = 1.$$

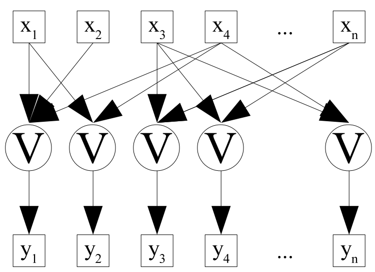
Pirmais, kas uzreiz ir redzams, ka augšējās un apakšējās robežas nesakrīt. Gadījumā, ja $k \geq 2$, tās atšķiras konstantu skaitu reižu (četras reizes, ja $k = 2$, vai mazāk, ja $k > 2$), bet, gadījumā, ja $k = 1$, atšķirība nav pat konstanta. Gribētos ticēt, ka, līdzīgi kā stāvokļu sarežģītības gadījumā, arī šeit ir spēka Šenona efekts. Bet tad jāatbild uz jautājumu, kurus no šiem novērtējumiem iespējams uzlabot?

Gadījumā, kad $k \geq 2$, atšķirība ir divās vietās: reizinātājos $k-1$ un k, kā arī dalītājā 2 apakšējā robežā. Pirmā atšķirība domājams nāk no 2.1 novērtējuma, tur arī redzams, ka augšējā robežā ir reizinātājs k, bet apakšējā — reizinātājs $k-1$. Līdz ar to, lai uzlabotu šo reizinātāju, jāuzlabo apakšējais novērtējums neekvivalentu GNA skaitam ar ne vairāk kā s stāvokļiem.

Savukārt dalītājs 2 domājams nāk no pārejas shēmas konstrukcijas. Šajā loģiskajā shēmā kritiskā vieta ir daudzās disjunkcijas, kas atbilst vairākām izejām un kā tās vienlaicīgi efektīvi konstruēt. Kā šī jautājuma esenci var aplūkot loģisko shēmu konstrukcijas uzdevumu funkciju klasei $f : \{0,1\}^n \to \{0,1\}^n$, kur katram i

$$y_i = x_{i_1} \vee x_{i_2} \vee x_{i_k},$$

šeit x_i un y_i ir attiecīgi ieejas un izejas mainīgie, katrs izejas mainīgais ir dažu ieejas mainīgo disjunkcija (piemēru skat. 5.3. zīm.). Apzīmēsim šo funkciju klasi ar F^n.



5.3. att. Loģiskās shēmas piemērs funkcijai no klases F^n.

Šai Būla funkcijai tās sarežģītības labākās zināmās augšējās un apakšējās robežas atšķiras 2 reizes.

5.6. Teorēma *Gandrīz visām Būla funkcijām $f \in F^n$:*

$$\frac{n^2}{2\log n} \lesssim C(f) \lesssim \frac{n^2}{\log n}.$$

Pierādījums Pierādījums augšējai robežai ir tieši tāds pats, kā 5.3. teorēmai, tāpēc to tikai īsi ieskicēsim: Sadala visus ieejas mainīgos grupās pa c, katrā no $\frac{n}{c}$ grupām izrēķina visas 2^c disjunkcijas, kam kopā nepieciešami aptuveni $\frac{n2^c}{c}$ loģiskie elementi (disjunkcijas). Katru izejas mainīgo tagad var izteikt kā disjunkciju no $\frac{n}{c}$ apakšformulām (katru no savas grupas). Tātad katram no tiem vajag $\frac{n}{c}$ disjunkcijas, kas kopā sastāda $\frac{n^2}{c}$ disjunkcijas. Tātad kopējais loģisko elementu (kas visi ir disjunkcijas) skaits šajā loģiskajā shēmā būs $\frac{n2^c}{c} + \frac{n^2}{c}$.

Ja izvēlas $c = \log n - \log\log n$, tad summā $2^c\frac{n}{c} + \frac{n^2}{c}$ dominē otrais loceklis un tas ir $\frac{n^2}{\log n}$.

Apakšējai robežai ievērosim, ka $|F^n| = 2^{n^2}$. Tā kā $\frac{n^2}{2\log n} \lesssim \frac{n^2}{2\log n} - n$, tad mums pietiks novērtēt to funkciju daļu ε_n klasē F^n, kuru loģiskās shēmas sarežģītība nepārsniedz $\frac{n^2}{2\log n} - n$ un pierādīt, ka $\varepsilon_n \to 0$, ja $n \to \infty$. Tā vietā pierādīsim, ka $\log\varepsilon_n \to -\infty$.

Funkciju skaits klasē F^n, kuru sarežģītība nepārsniedz $\frac{n^2}{2\log n} - n$, noteikti nepārsniedz visu funkciju skaitu ar šādu pat sarežģītību, tāpēc

$$\varepsilon_n \leq \frac{N(n, n, \frac{n^2}{2\log n} - n)}{2^{n^2}},$$

šeit tāpat kā 2.2. teorēmā $N(n, m, c)$ ir dažādo Būla funkciju skaits ar n ieejas mainīgajiem un m izejas mainīgajiem, kuru sarežģītība nepārsniedz c.

No 2.2. teorēmas izriet, ka

$$\begin{aligned}\log\varepsilon_n &\leq \log\frac{\left(\frac{n^2}{2\log n}\right)^{\left(\frac{n^2}{2\log n}\right)} 9^{\left(\frac{n^2}{2\log n}\right)}}{2^{n^2}} = \\ &= \left(\frac{n^2}{2\log n}\right)(2\log n + \log 9 - 1 - \log\log n) - n^2 = \\ &= \left(\frac{n^2}{2\log n}\right)(\log 9 - 1 - \log\log n) \to -\infty. \quad \blacksquare\end{aligned}$$

Vai kādu no šiem novērtējumiem var uzlabot ir sarežģīts jautājums, lai arī pats uzdevums neliekas pārāk grūts. Bet to, ka tas nav triviāls, var ilustrēt ar faktu, ka dažām funkcijām no F^n optimālās loģiskās shēmas satur arī konjunkcijas (lai arī pietiktu tikai ar disjunkcijām)[37].

Unāro valodu gadījumā var pārliecināties, ka pat 2.1. teorēmas augšējās robežas ievietošana 5.4. teorēmas aprēķinos palielinās tās apakšējo robežu tikai līdz s. Tas pieļauj iespēju, ka optimālu loģisko shēmu konstrukcijas unāriem GNA vēl nav izsmeltas. Tāpēc tas, vai 5.3. teorēmas konstrukciju unāriem GNA ir iespējams uzlabot, paliek kā jautājums turpmākai izpētei.

6. nodaļa

Valodu operācijas

Šajā nodaļā aplūkosim kā mainās BC-sarežģītība pie dažādām valodu operācijām: apvienojuma, šķēluma, konkatenācijas, Klīni slēguma un apvēršanas, kā arī salīdzināsim to ar attiecīgo stāvokļa sarežģītību.

Stāvokļu sarežģītības pie valodu operācijām sistemātiska izpēte tika sākta salīdzinoši vēlu, tikai 90-os gados, kad tika iegūti sakrītoši augšējie un apakšējie novērtējumi svarīgākajām valodu operācijām[46]. Turpat arī tika pamatots, ka sarežģītība valodu operācijām ir atkarīga no tā, vai mēs aplūkojam tās viena simbola alfabētā vai vairāku, izrādās arī, ka galīgām valodām tā ir cita (mazāka) nekā vispārīgajā gadījumā[10].

Šī joma turpina attīstīties, mūsdienās tiek aplūkotas gan kompleksas operācijas (Klīni slēgums no valodu apvienojuma u.c.)[34], gan tā tiek novērtēta dažādām specifiskām valodu klasēm[16] vai nedeterminētiem automātiem[18]. Īsu pārskatu var atrast, piemēram, rakstā [45].

Vispirms aplūkosim klasiskus rezultātus, kā valodu operācijas ietekmē stāvokļu sarežģītību, un pēc tam pāriesim pie BC-sarežģītības.

Pieņemsim, ka mums ir dotas divas valodas L_1 un L_2 ar stāvokļu sarežģītību, attiecīgi, m un n. GDA $A_1(Q_1, \Sigma, \delta_1, q_0^1, \tilde{Q_1})$ ir minimālais GDA, kas pazīst valodu L_1 ($|Q_1| = m$) un GDA $A_2(Q_2, \Sigma, \delta_2, q_0^2, \tilde{Q_2})$ ir minimālais GDA, kas pazīst valodu L_2 ($|Q_2| = n$).

6.1. Teorēma *[33]*

- *Ja* $L_3 = L_1 \cup L_2$ *vai* $L_3 = L_1 \cap L_2$ *tad* $sc(L_3) \leq mn$.
- *Ja* $L_3 = L_1^R$, *tad* $sc(L_3) \leq 2^m$.
- *Ja* $L_3 = L_1 L_2$, *tad* $sc(L_3) \leq m2^n - 2^{n-1}$.
- *Ja* $L_3 = (L_1)^*$, *tad* $sc(L_3) \leq 2^{m-1} + 2^{m-2}$.

Pierādījums Tā kā šie rezultāti ir labi zināmi, tad to pierādījumu tikai ieskicēsim.

Visos šajos gadījumos var uzkonstruēt GDA, kas pazīst L_3 un kura stāvokļu telpa ir atkarīga no izejas GDA stāvokļu telpām (telpas). Apvienojuma un šķēluma gadījumā šī telpa būs $Q_1 \times Q_2$, apvērstās valodas un Klīni slēguma gadījumā — 2^{Q_2}, bet konkatenācijas gadījumā: $Q_1 \times 2^{Q_2}$. Apvienojuma un šķēluma gadījumā GDA A_1 un A_2 darbojas neatkarīgi, apvērstās valodas gadījumā mēs varam visas pārejas pavērst pretējā virzienā, iegūstot GNA, ko pēc tam vajag determinizēt. Klīni slēguma gadījumā no visiem beigu stāvokļiem jāpievieno ε pārejas uz sākuma stāvokli, kas doto GDA pārvērš par GNA, ko pēc tam var determinizēt. Konkatenācijas gadījumā paralēli darbojas A_1 kā GDA un A_2 kā GNA, pie tam vienmēr, kad A_1 atrodas kādā beigu stāvoklī A_2 vēl papildus tiek uzstādīts sākuma stāvoklī.

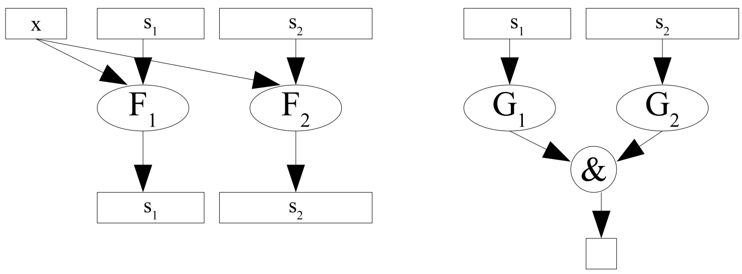
Teorēmas stāvokļu skaita novērtējumi sakrīt (vismaz ar kārtu) ar augstākminētajiem stāvokļu telpas izmēriem, bet konkatenācijas un Klīni slēguma gadījumā papildus var identificēt arī dažus nesasniedzamus stāvokļus, līdz ar ko šīm operācijām augšējo novērtējumu var nedaudz uzlabot. ∎

Visi šie novērtējums stāvokļu sarežģītībai ir optimāli, visām šīm operācijām var piemeklēt valodas (maksimums 3 simbolu alfabētā), kuru stāvokļu sarežģītība sasniedz šo augšējo robežu.

Aplūkosim tagad, kā BC-sarežģītība mainās līdz ar visām šīm valodu operācijām. Visām operācijām pieņemsim, ka mums dotas divas valodas L_1 un L_2, kurām $m = sc(L_1)$, $n = sc(L_2)$, $a = C_{\text{BC}}(L_1)$, $b = C_{\text{BC}}(L_2)$, $k = |\Sigma|$. Sāksim ar valodu apvienojumu un šķēlumu.

6.2. Teorēma *Ja* $L_3 = L_1 \cup L_2$ *vai* $L_3 = L_1 \cap L_2$ *tad* $C_{\text{BC}}(L_3) \le a + b + 1$.

Pierādījums Pieņemsim, ka loģiskās shēmas (F_1, G_1) reprezentē GDA, kas atpazīst L_1 ar minimālo BC-sarežģītību, un loģiskās shēmas (F_2, G_2) reprezentē GDA, kas pazīst L_2 (arī ar minimālo BC-sarežģītību), stāvokļu bitu skaits tām ir, attiecīgi, b_Q^1 un b_Q^2. Pārejas shēma GDA, kas pazīst L_3, sastāvēs no loģiskajām shēmām F_1 un F_2, kuras strādās paralēli (6.1. zīm. pa kreisi). Akceptēšanas shēma sastāvēs no loģiskajām shēmām G_1 un G_2, kas katra darbosies uz sev atbilstošajiem ieejas mainīgajiem (stāvokļu bitiem) un kuru rezultāti tiks savienoti ar disjunkciju (apvienojumam) vai konjunkciju (šķēlumam), skat. 6.1. zīm. pa labi.



6.1. att. Pārejas shēma (pa kreisi) un akceptēšanas shēma (pa labi) valodu šķēlumam

Šādas loģiskās shēmas reprezentācijas sarežģītība ir

$$C_{\text{BC}}(L_3) \le C(F_1) + C(F_2) + C(G_1) + C(G_2) + 1 + b_Q^1 + b_Q^2 = a + b + 1. \qquad \blacksquare$$

Vārds $x_1 x_2 \ldots x_n$ pieder apvērstajai valodai L_1^R tad un tikai tad, ja $x_n \ldots x_2 x_1$ pieder valodai L_1. GNA N, kas atpazīst L_1^R, var iegūt no GDA A_1, kas atpazīst L_1, uzstādot par N sākuma stāvokļiem visus A beigu stāvokļus, uzstādot q_0, par N vienīgo beigu stāvokli un apgriežot visas pārejas pretējā virzienā. GDA, kas atpazīst L_1^R, var iegūt determinizējot šo GNA N.

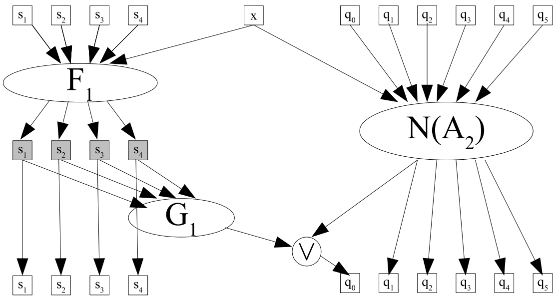
6.3. Teorēma $C_{BC}(L_1^R) \le (2k+4)m$

Pierādījums Šī teorēma izriet tiešā veidā no 5.1. teorēmas un tā, ka GNA N ir tieši tikpat pāreju, cik bija GDA A_1 — tātad km. Vēl papildus ne vairāk kā $3m$ loģisko elementu jāierēķina iespējamajai sākuma stāvokļa uzstādīšanai uz nullēm. ∎

Valoda L_1L_2, kas ir valodu L_1 un L_2 konkatenācija, sastāv no visiem vārdiem uw, tādiem, ka $u \in L_1$ un $w \in L_2$.

6.4. Teorēma $C_{\text{BC}}(L_1L_2) \le a + (2k+1)n$

Pierādījums Pieņemsim, ka GDA A_1 pazīst L_1 ar minimālo BC-sarežģītību, bet GDA A_2 pazīst L_2 ar minimālo stāvokļu sarežģītību. Konstrukcija būs tāda pati kā stāvokļu sarežģītības gadījumā. Paralēli darbosies A_1 kā GDA un A_2, kā GNA, un vienmēr, kad A_1 būs kādā no saviem beigu stāvokļiem, A_2 papildus saviem tekošajiem stāvokļiem nonāks arī sākuma stāvoklī.



6.2. att. Pārejas shēma F GDA, kas pazīst valodu konkatenāciju L_1L_2

Pārejas shēmu GDA, kas pazīst L_1L_2 konstruē sekojoši (6.2. zīm.): tās stāvokļu biti sastāvēs no A_1 kodējuma stāvokļu bitiem un A_2 stāvokļiem. Pārejas shēma F_1 sastāvēs no A_1 pārejas shēmas un A_2 kā GNA pārejas shēmas (kā 5.1. teorēmā),kuras darbosies paralēli, šo pēdējo apzīmēsim ar $N(A_2)$. Katrā solī A_1 akceptēšanas shēma G_1noskaidros, vai gadījumā A_1 neatrodas beigu stāvoklī, un, ja tā, tad papildus uzstādīs $N(A_2)$ arī tās sākuma stāvoklī q_0.

No 5.1. teorēmas izriet, ka $C(N(A_2)) \le t + (k+1)n$ un, tā kā A_2 ir GDA, tad, $t = kn$.

Akceptēšanas shēma pārbaudīs to, ka A_2 atrodas akceptējošā stāvoklī, tam nepieciešama ne vairāk kā $n-1$ disjunkcija. Stāvokļa bitu skaits šai loģiskās shēmas reprezentācijai ir $b_Q^1 + n$, kur b_Q^1 ir stāvokļa bitu skaits A_1 reprezentācijā (6.2. zīmējumā $b_Q^1 = 4$ un $n = 6$).

Tātad kopējā BC-sarežģītība GDA, kas pazīst L_1L_2 nepārsniedz

$$C_{\text{BC}}(L_1L_2) \le C(F_1) + C(G_1) + C(N(A_2)) + 1 + n - 1 + b_Q^1 + n \le$$
$$\le a + kn + (k+1)n + 2n = a + (2k+3)n. \quad ∎$$

6.5. Teorēma $C_{\text{BC}}(L_1^*) \le (3k+1)m.$

Pierādījums GNA, kas pazīst L_1^*, var iegūt no GDA, kas pazīst L_1, pievienojot papildus katrai pārejai, kas pāriet uz kādu no beigu stāvokļiem, arī pāreju uz sākuma stāvokli. Pāreju skaits šim GNA nepārsniegs $2km$. Konstruējot loģiskās shēmas reprezentāciju šim GNA, kā 5.1. teorēmā, iegūst prasīto. ∎

Operācija	Stāvokļu sarežģītība	BC-sarežģītība
$L_1 \cup L_2$	mn	$a+b+1$
$L_1 \cap L_2$	mn	$a+b+1$
L^R	2^m	$(2k+4)m$
$L_1 L_2$	$m2^n - 2^{n-1}$	$a+(2k+3)n$
L_1^*	$2^{m-1} + 2^{m-2}$	$(3k+1)m$

6.1. tabula. Stāvokļu sarežģītība un BC-sarežģītība valodu operācijām

Tabulā 6.1 ir salīdzināta stāvokļu sarežģītība un BC-sarežģītība dažādām valodu operācijām.

Iegūtajos rezultātos var ievērot, ka visām operācijām iegūtās valodas BC-sarežģītība ir daudz mazāka par tās maksimālo iespējamo vērtību, īpaši izteikti tas ir redzams tām operācijām, pie kurām stāvokļu skaits var pieaugt eksponenciāli. Piemēram, konkatenācijas gadījumā, gandrīz visām valodām, kuru stāvokļu sarežģītība ir $m2^n - 2^{n-1}$, to BC-sarežģītība ir apmēram $(k-1)(m2^n - 2^{n-1})$ (4.11. teorēma), kas ir daudz lielāka nekā maksimālā BC-sarežģītība tām valodām, kas iegūtas valodu konkatenācijas rezultātā ($a + (2k+3)n$).

7. nodaļa

BC-sarežģītība un GDA minimizācija

Ja mēs atgriežamies pie klasiskā stāvokļu kodēšanas uzdevuma, tad tas ir cieši saistīts ar GDA stāvokļu minimizāciju. Lai arī pirmajā brīdī varētu šķist, ka šīs lietas ir neatkarīgas: doto GDA mēs varam minimizēt un tad mēģināt atrast tam optimālu stāvokļu kodējumu, tā nebūt nav, ne vienmēr minimālo GDA varēs praktiski realizēt tikpat efektīvi kā sākotnējo.

Jau 1962. gadā Hartmanis un Stērns[17] aplūkoja 8 stāvokļu GDA, kura realizācijai pietiek ar 19 diodēm, bet, ja to minimizē līdz 7 stāvokļiem, tad tam vajadzīgas jau 22 diodes (tā nav apakšējā robeža, bet labākā autoriem zināmā realizācija). Arī citi autori uzsver, ka stāvokļu minimizāciju nevajag aplūkot atsevišķi no stāvokļu kodēšanas uzdevuma, bet šie jautājumi jārisina sinhroni[8].

Šajā nodaļā mēs aplūkosim šo jautājumu no teorētiskā — BC-sarežģītības viedokļa un pamatosim, ka minimālā GDA BC-sarežģītība var atšķirties no sākotnējā GDA BC-sarežģītības ne tikai ”par 3 diodēm”, bet arī krietni vien vairāk.

Izrādās, ka minimālā GDA BC-sarežģītība nav polinomiāli ierobežota ar sākotnējā (neminimizētā) GDA BC-sarežģītību. Tas nozīmē, ka stāvokļu skaita minimizācija var stipri palielināt GDA iekšējās struktūras sarežģītību. No otras puses — tas nozīmē to, ka dažos gadījumos, ekvivalentu stāvokļu izmantošana automātā, var samazināt to strukturālo sarežģītību.

Atcerēsimies, ka ar $M(A)$ ($M(L)$) apzīmē minimālo GDA, kas ir ekvivalents A (kas pazīst L).

7.1. Teorēma *Ja eksistē tāds polinoms $p(x)$, ka visām regulārām valodām L binārā alfabētā izpildās $C_{\text{BC}}(M(L)) < p(C_{\text{BC}}(L))$, tad PSPACE$\subseteq$ P/poly.*

Pierādījums Pierādījuma galvenā ideja ir sekojoša: dotai valodai $V \in$ PSPACE konstruēsim GDA A_n^V, kas strādā binārā alfabētā un akceptē vārdu w, ja tā pirmie n simboli pieder valodai V, bet dara to pēc eksponenciāli ilga laika (pārējos ieejas simbolus līdz tam automāts ignorē). Šādu automātu var konstruēt ar polinomiālu BC-sarežģītību attiecībā pret n, modelējot tā stāvokļu telpā Tjūringa mašīnas, kas pazīst V, darbu.

Bet atbilstošais minimālais GDA $M(A_n^V)$ ”zinās”, vai vārds jāakceptē jau pēc pirmo n simbolu nolasīšanas — tas atradīsies vienā no diviem iespējamajiem stāvokļiem, atkarībā no tā vai $w_1 w_2 \ldots w_n \in V$. Ja $C_{\text{BC}}(M(A_n^V))$ ir polinomiāli atkarīga no

$C_{\mathtt{BC}}(A_n^V)$, tad arī tā ir polinomiāla attiecībā pret n, bet no $M(A_n^V)$ loģiskās shēmas reprezentācijas tad savukārt var konstruēt polinomiāla izmēra (attiecībā pret n) loģisko shēmu, kas nosaka vai vārds pieder V. Tas nozīmē, ka $V \in$ PSPACE $\Rightarrow V \in$ P/poly, kas arī bija vajadzīgs.

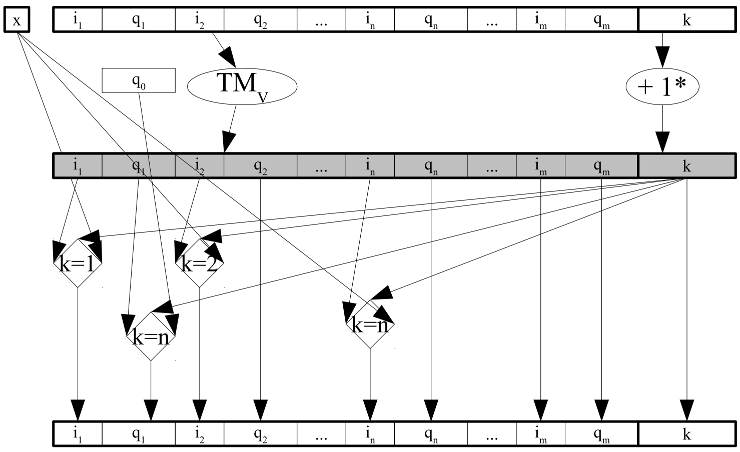
Un tagad vēlreiz tas pats mazliet sīkāk un precīzāk. Pieņemsim, ka eksistē tāds polinoms p, ka jebkurai regulārai valodai L bināră alfabētā $\Sigma = \{0,1\}$ izpildās $C_{\mathtt{BC}}(M(L)) < p(C_{\mathtt{BC}}(L))$. Ņemsim patvaļīgu valodu $V \in$ PSPACE (binārā alfabētā). Mums jāpierāda, ka eksistē tāds polinoms $r(n)$, ka jebkuram n var uzkonstruēt loģisko shēmu ar ne vairāk kā $r(n)$ loģiskajiem elementiem, kas vārdiem garumā n nosaka, vai tie pieder V.

No 2.9. teorēmas izriet, ka eksistē tāda Tjūringa mašīna TM_V, kas pazīst V, lietojot ne vairāk ka $q(n)$ lentes rūtiņas ne vairāk kā $2^{q(n)}$ soļos, kur $q(n)$ ir kāds polinoms. Apzīmēsim $m = q(n)$ un aplūkosim regulāru valodu L_n^V binārā alfabētā, kurai pieder tie vārdi garumā $n+2^m$, kuru pirmo n simbolu veidotais vārds pieder V:

$$w_1 w_2 \ldots w_t \in L_n^V \iff t = n + 2^m \text{ un } w_1, \ldots, w_n \in V.$$

Vispirms konstruēsim GDA A_n^V, kurš pazīst L_n^V un kura BC-sarežģītība ir polinomiāli atkarīga no n. Automāta A_n^V stāvokļa reģistrā vajadzēs modelēt Tjūringa mašīnu TM_V kā arī iekļaut skaitītāju, kas prot skaitīt līdz $n + 2^m$.

Automāta A_n^V darbības plāns ir šāds: vispirms ielasīt pirmos n simbolus stāvokļu reģistrā un tad sākt simulēt TM_V darbu, paralēli tam visam skaitot līdzi soļu (ielasīto simbolu) skaitu. Pēc $2^m + n$ simbolu nolasīšanas TM_V būs veikusi 2^m soļus un tātad noteikti būs apstājusies un ierakstījusi atbildi (1, ja $x_1, \ldots, x_n \in V$, 0, ja nē) savas lentes pirmajā rūtiņā. Lai noskaidrotu, vai vārds pieder L_n^V, jāpārbauda nolasīto simbolu skaits (pēc skaitītāja) un, ja tas sakrīt ar $2^m + n$, tad jāskatās, kas rakstīts lentes pirmajā rūtiņā.



7.1. att. Pārejas shēma F automātam A_n^V

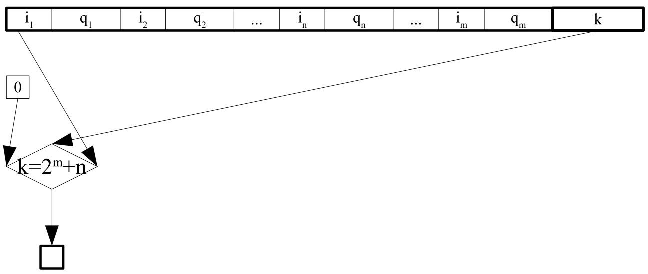
GDA A_n^V pārejas shēma F redzama 7.1. zīmējumā. Augšējā rindiņā x ir ieejas bits (ar dabīgu kodējumu), bet pārējie ir stāvokļa biti: $i_1 q_1, \ldots, i_m q_m$ ir TM_V lentes un stāvokļa bloki, kuros tiek simulēta tās darbība (kā 2.6. teorēmā), bloks k ir $m+1$ bita skaitītājs. Pelēki iekrāsotās rūtiņas satur starprezultātus, kas izdalīti atsevišķi, lai labāk paskaidrotu loģiskās shēmas darbību.

Izvēlēsimies tādu Tjūringa mašīnas lentes kodējumu, ka tukšais simbols λ tiek kodēts ar 00, nulle tiek kodēta ar 01, vieninieks tiek kodēts ar 11. Stāvokļa kodējumam svarīgi, lai tukšais stāvoklis $\tilde{q}$ tiktu kodēts ar visām nullēm, pārējie stāvokļi var tikt kodēti patvaļīgi. Šāds kodējums nodrošinās to, ka sākumā uz lentes ir visas nulles, kā jābūt GDA loģiskās shēmas reprezentācijā, kā arī to, ka, lai noskaidrotu, kāds simbols (0 vai 1) atrodas TM_V lentes pirmajā rūtiņā i_1, pietiks paskatīties uz A_n^V stāvokļa reģistra pirmo bitu. Arī skaitītāja sākuma vērtība ir nulle.

Pārejas shēma sastāv no 3 veidu komponentēm. "TM_V" komponente simulē Tjūringa mašīnas TM_V darbību, "+1*" komponente pieskaita vieninieku skaitītājam, līdz tas ir sasniedzis savu maksimālo vērtību $2^{m+1} - 1$, pēc kā tā vērtība paliek konstanta (līdz ar to tā nav klasiska vieninieka pieskaitīšana 2.5. teorēmas izpratnē, bet atšķiras tikai vienai vērtībai — visiem vieniniekiem). Atgādināsim, ka izvēles elementi "$k = j$", kurus apzīmē ar rombiņiem, atgriež savu kreiso ieeju, ja nosacījums neizpildās, un labo, ja izpildās, dati, kas nepieciešami paša nosacījuma izrēķināšanai, tiek padoti rombiņa vidū. Visi izvēles elementi, kuri saņem k vērtību nosacījumam no skaitītāja, saņem to pēc vieninieka pieskaitīšanas — līdz ar to nosacījums $k = j$ izpildās tieši j-ajā solī (kad tiek nolasīts j-tais simbols no lentes).

Pirmajos n soļos reģistri $i_1, \ldots, i_n$ viens pēc otra tiek aizpildīti ar ieejas simbolu $x_1, \ldots, x_n$ iekodētām vērtībām. Tas tiek darīts ar izvēles elementu "$k = j$" palīdzību, kuri ieraksta doto ieejas bitu i_j reģistrā tad un tikai tad, ja $k = j$. Kopā tātad ir n šādi izvēles elementi.

Pēc n simbolu nolasīšanas izvēles elements $k = n$ uzstāda q_1 reģistru uz TM_V sākuma stāvokli q_0, tas nozīmē, ka TM galviņa tagad atrodas pirmajā rūtiņā un pati TM atrodas sākuma stāvoklī. Līdz tam visi stāvokļu bloki q_j bija uzstādīti uz $\tilde{q}$ lidz ar to TM nekādas darbības neizveica. Bet tagad TM_V simulēšana sākas (un turpinās visu laiku).



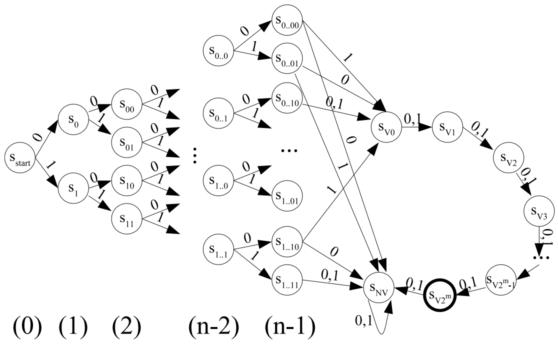
7.2. att. Akceptēšanas shēma automātam A_n^V

Pēc 2^m soļiem TM_V ir apstājusies — tas nozīmē, ka tās rezultāts (vai $x_1, \ldots, x_n$ pieder V) ir ierakstīts reģistrā i_1. Akceptēšanas shēma G (7.2. zīm.) akceptē ieejas vārdu tad un tikai tad, ja $k = 2^m + n$ un i_1 reģistrs satur iekodētu vieninieku (tā pirmais bits ir 1).

Novērtēsim tagad A_n^V BC-sarežģītību. TM simulācijai nepieciešami $m(2 + \lceil \log(|Q| + 1) \rceil)$ stāvokļa biti: divi biti katram nokodētam datu bitam i_j un $\lceil \log(|Q| + 1) \rceil$ biti katram nokodētam TM_V stāvoklim q_j, skaitītājam nepieciešams $m + 1$ bits, līdz ar to kopējais stāvokļa bitu skaits būs $b_Q = (1 + m) + m(2 + \lceil \log(|Q| + 1) \rceil)$, kur $|Q|$ ir TM_V stāvokļu skaits.

No 2.6. teorēmas izriet, ka TM_V komponentes sarežģītība ir cm, kur c ir konstante, kas atkarīga no TM_V, bet nav atkarīga no n (tātad arī no m). Katras pārbaudes $k = j$ sarežģītība ir lineāra attiecībā pret n, tātad to kopējā sarežģītība ir kvadrātiska attiecībā pret n. Skaitītāja bloka sarežģītība ir lineāra attiecībā pret m. Arī akceptēšanas shēmas sarežģītība acīmredzami ir lineāra attiecībā pret m.

Tā kā m ir polinomiāli atkarīgs no n, tad GDA A_n^V BC-sarežģītība ir polinomiāla attiecībā pret n: $C_{\text{BC}}(A_n^V) \leq h(n)$, kur $h(n)$ ir kāds polinoms.

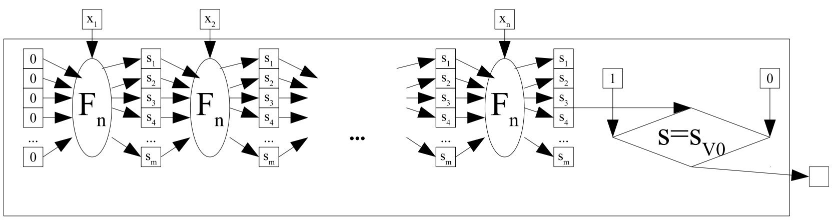


7.3. att. Automāta $A_n'^V$ stāvokļu pāreju diagramma

Aplūkosim tagad savādāk konstruētu GDA $A_n'^V$, kurš arī pazīst valodu L_n^V, tā stāvokļu pārejas diagramma shematiski attēlota 7.3. zīmējumā.

Pirmajos n soļos ieejas simboli tiek ielasīti un saglabāti "stāvokļu atmiņā", katrā kolonnā $1 \leq i \leq n-1$ ir 2^i stāvokļu visām ieejas virknēm garumā i. Kad tiek nolasīts n-tais ieejas simbols, GDA pāriet vai nu uz stāvokli S_{V0} vai S_{NV} atkarībā no tā, vai $x_1 \dots x_n \in V$ vai $x_1 \dots x_n \notin V$. Šis solis stāvokļu diagrammā ir parādīts tikai shematiski, tas ir atkarīgs no valodas V. Stāvoklis S_{NV} ir visu noraidošs, bet no stāvokļa s_{V0} automāts pēc 2^m stāvokļiem nonāk vienīgajā akceptējošajā stāvoklī s_{V2^m} (neatkarīgi no ieejas simboliem), no kura tas tālāk pāriet uz stāvokli s_{NV}, kurā tas paliek uz visiem laikiem.

Var redzēt, ka pēc n simbolu nolasīšanas GDA $A_n'^V$ jau satur viegli pārbaudāmu informāciju par to, vai $x_1, \dots, x_n \in V$ — tas atrodas tieši vienā no diviem stāvokļiem s_{V0} vai s_{NV} atkarībā no tā, vai $x_1, \dots, x_n \in V$ vai ne. To var izmantot, lai uzkonstruētu loģisko shēmu valodas V vārdiem garumā n, līdzīgi, kā tas tika darīts 4.9. teorēmas pierādījumā.



7.4. att. Loģiskā shēma, kas pazīst valodas V vārdus garumā n

Lai arī $A_n'^V$ iespējams nav minimālais automāts (var gadīties, ka tā "kreiso pusi" var minimizēt), tā minimizācija var novest tikai pie dažu stāvokļu "salipšanas" kreisajā pusē un arī minimālajam GDA $M(A_n'^V) = M(L_n^V)$ atradīsies divi stāvokļi s_{V0} un s_{NV}, kuros tas atradīsies pēc n simbolu nolasīšanas un pēc kuriem varēs noteikt vai $x_1, \dots, x_n \in V$.

No teorēmas pieņēmuma $C_{\text{BC}}(M(A_n^V)) < p(C_{\text{BC}}(A_n)) < p(h(n))$ līdz ar to eksistē automāta $M(A_n'^V)$ loģiskās shēmas reprezentācija (F_n,G_n) ar b_n stāvokļu bitiem, kuras BC-sarežģītība ir polinomiāla attiecībā pret n.

Izmantojot to, konstruēsim loģisko elementu shēmu, kas pazīst valodas V vārdus garumā n. Savienosim n pārejas shēmas F_n kopā (7.4. zīm.), lai iegūtu loģisko shēmu, kas modelē automāta $M(A_n'^V)$ pirmos n soļus. Galā pievienosim izvēles elementu "$s = s_{V0}$" (tā sarežģītība nepārsniedz $2b_n$) un atgriezīsim 1 vai 0 atkarībā no tā vērtības. Tā kā pēc n simbolu nolasīšanas $M(A_n)$ atrodas stāvoklī s_{V0} tad un tikai tad, ja $x_1 \dots x_n \in V$, tad šī loģisko elementu shēma pazīst valodas V vārdus garumā n. Tās sarežģītība (ja $n \geq 2$) ir

$$r(n) = n \cdot C(F_n) + C(s = s_{V0}) \leq n \cdot C(F_n) + 2b_n \leq n \cdot C_{\text{BC}}((F_n, G_n)) \leq np(h(n)),$$

kas ir polinomiāla attiecībā pret n.

Tā kā katrai valodai $V \in$ PSPACE šādi var uzkonstruēt polinomiāla izmēra loģisko elementu shēmu jebkuram ieejas datu garumam n, tad $V \in$ PSPACE$\Rightarrow V \in$ P/poly. ∎

Karpa-Liptona teorēma apgalvo, ka, ja NP$\subseteq$P/Poly, tad polinomiālā hierarhija kolapsē līdz tās otrajam līmenim Σ_P^2 (skat. 2.6. nodaļu). Ir vispārzināms fakts, ka, ja PSPACE $\subseteq$ P/poly, tad PSPACE$\subseteq \Sigma_2^P \cap \Pi_2^P$. Tāpēc, lai arī nav pierādīts, ka PSPACE$\not\subseteq$ P/poly, ir vispārpieņemts uzskatīt, ka šis apgalvojums ir patiess. No kā izriet, ka regulāras valodas minimālā GDA BC-sarežģītība nav polinomiāli ierobežota attiecībā pret tās BC-sarežģītību.

8. nodaļa

BC-sarežģītība galīgiem automātiem ar izeju

Kā viena no motivācijām šim darbam ir galīgu automātu izmantošana praksē un ar to saistītā stāvokļu kodēšanas problēma. Praksē šo jautājumu visbiežāk aplūko tieši galīgiem automātiem ar izeju, un arī autora pirmajā rakstā par šo tēmu[40] BC-sarežģītība tika aplūkota tieši automātiem ar izeju. Tiem nodefinēt loģiskās shēmas sarežģītību ir kaut kāda ziņā pat vienkāršāk, nav jādomā, ko iesākt ar akceptējošo stāvokļu kopu. Lai arī turpmākajos rakstos[39][41][42] un līdz šim šajā darbā tika aplūkoti tikai galīgi automāti bez izejas (galvenokārt tas tika darīts, lai varētu runāt par BC sarežģītību regulārām valodām), ir vērts vismaz īsi aplūkot arī automātus ar izeju.

Gandrīz visi rezultāti, kas ir spēkā galīgiem automātiem bez izejas, ir spēkā arī galīgiem automātiem ar izeju (Šenona efekts, ...) un pierādījumi arī ir tādi paši, tāpēc šajā nodaļā tikai īsi tos uzskaitīšu un arī pierādījumos norādīšu tikai atšķirības no attiecīgās teorēmas pierādījuma galīgiem automātiem bez izejas.

Galīgi automāti ar izeju ir galīgo automātu variācija, kas katru vārdu tā vietā, lai to akceptētu vai noraidītu, pārveido par citu tikpat garu vārdu. Katrā solī šāds automāts lasa vienu simbolu no ieejas lentes un atkarībā no tā un no tekošā stāvokļa raksta simbolu izejas lentē, un pāriet uz nākamo stāvokli. Formāli to var aprakstīt šādi:

Definīcija Galīgs determinēts automāts ar izeju ir kortežs $(Q, X, Y, \delta, \sigma, q_0)$, kur

1. Q ir stāvokļu telpa (galīga kopa)
2. X ir ieejas alfabēts (galīga kopa)
3. Y ir izejas alfabēts (galīga kopa)
4. $\delta : X \times Q \to Q$ ir stāvokļu pārejas funkcija
5. $\sigma : X \times Q \to Y$ ir izejas funkcija
6. $q_0 \in Q$ ir sākuma stāvoklis

Galīgs automāts ar izeju sāk darbu stāvoklī q_0 un katrā solī nolasa no ieejas lentes vienu simbolu, raksta izejas simbolu uz izejas lentes un nomaina savu stāvokli.

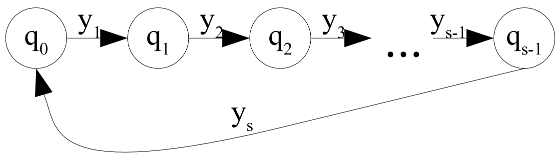
Ja tas atrodas stāvoklī $q \in Q$ un nolasa ieejas simbolu $x \in \Sigma$, tad tas uz izejas lentes raksta $\sigma(x, q)$ un pāriet uz jaunu stāvokli $\delta(x, q)$.

Šādi galīgs automāts ar izeju pārveido vārdus garumā n no ieejas alfabēta X par vārdiem garumā n izejas alfabētā Y. Ekvivalenci un minimizāciju galīgiem automātiem ar izeju definē tāpat kā akceptoriem. Gadījums, kad $|Y| = 1$, ir triviāls, visi automāti ar šādu izejas alfabētu, darbojas vienādi, tāpēc turpmāk visur pieņemsim, ka $|Y| \geq 2$.

Novērtēt galīgu automātu ar izeju skaitu ir sarežģītāk, jo literatūrā šāds novērtējums neatradās, bet novērtējums akceptoriem[14] izrādījās viegli adaptējams mūsu vajadzībām, īpaši ņemot vērā to, ka 4. nodaļas asimptotiskajiem novērtējumiem pietiek ar salīdzinoši rupju novērtējumu.

8.1. Teorēma ([14]) *Neekvivalentu minimālu galīgu automātu ar izeju skaits k-simbolu ieejas alfabētā X ar ne vairāk kā s stāvokļiem un izejas alfabētu Y ($|Y| \geq 2$) nav mazāks par $|Y|^s s^{(k-1)s}$.*

Pierādījums Konstruēsim neekvivalentu automātu ar izeju apakškopu, kuras izmērs nav mazāks par $|Y|^s s^{(k-1)s}$. Izvēlēsimies patvaļīgu ieejas simbolu $a \in \Sigma$, un tam stāvokļu pārejas definēsim "pa apli" (8.1. zīm.). Katrai pārejai izejas simbolu varam izvēlēties $|Y|$ veidos, līdz ar to mums ir $|Y|^s$ šādu viena ieejas simbola automātu, nekādi divi no kuriem nav ekvivalenti (lai arī ne visi ir minimāli).



8.1. att. Dažādu minimālo automātu ar izeju konstruēšana gadījumā, kad $k = 1$, uz bultiņām rakstīts izejas simbols.

Visiem pārējiem $k-1$ ieejas simboliem stāvokļu pārejas un izejas simbolus definēsim patvaļīgi, to var izdarīt $(|Y| \cdot s)^{(k-1)s}$ veidos. Tātad kopējais šādu automātu skaits ir $|Y|^s(|Y| \cdot s)^{(k-1)s} \geq |Y|^s s^{(k-1)s}$.

Nekādi divi no šiem automātiem nav ekvivalenti. Ja mēs katram no tiem aplūkojam minimālo automātu, tad mēs iegūsim minimālu automātu kopu, katram no kuriem ir s vai mazāk stāvokļu, un kuru skaits ir vismaz $|Y|^s s^{(k-1)s}$. ∎

Galīgiem automātiem ar izeju atkrīt problēma ar atsevišķu loģisko elementu shēmu beigu stāvokļa pārbaudei. Tā kā gan izejas simbols, gan nākamais stāvoklis ir atkarīgi no vieniem un tiem pašiem argumentiem (no ieejas simbola un stāvokļa), tad to aprēķināšanu var veikt vienā loģiskajā shēmā. Šajā gadījumā šī shēma saņems ieejā nokodētu stāvokli un ieejas simbolu un atgriezīs iekodētu (nākamo) stāvokli un izejas simbolu.

Definīcija Par galīga automāta ar izeju A kodējumu $E(A)$ sauc kortežu (f_X, f_Y, f_Q), kas sastāv no ieejas alfabēta kodējuma f_X, izejas alfabēta kodējuma f_Y un stāvokļu telpas kodējuma f_Q pie tam $f_Q(q_0) = 0^{b_Q}$.

Definīcija Par dota galīga automāta ar izeju $A(Q,X,Y,\delta,\sigma,q_0)$ loģiskās shēmas reprezentāciju pie kodējuma $E(A) = (f_X, f_Y, f_Q)$ sauc loģisko elementu shēmu F, ja

- F ir $b_X + b_Q$ ieejas mainīgie un $b_Y + b_Q$ izejas mainīgie,
- visiem $x \in X$ un $q \in Q$, ja $q' = \delta(x, q)$ un $y = \sigma(x, q)$, tad $f_Y(y) f_Q(q') = F(f_X(x), f_Q(q))$,

Galīgam automāta ar izeju kodējums ir minimāls, ja visi trīs: stāvokļa, ieejas un izejas kodējumi ir minimāli.

Definīcija Galīga automāta ar izeju loģiskās shēmas reprezentācijas F BC-sarežģītība ir loģiskās shēmas F sarežģītība, kurai pieskaitīts stāvokļa bitu skaits:

$$C_{\text{BC}}(F) = C(F) + b_Q$$

Definīcija Galīga automāta ar izeju BC-sarežģītība pie kodējuma $E(A)$ ir minimālā sarežģītība, kāda ir kādai tā loģiskās shēmas reprezentācijai pie šī kodējuma:

$$C_{\text{BC}}(A, E(A)) = \min\{C(F) : F \text{ ir } A \text{ loģiskās shēmas reprezentācija pie kodējuma } E(A)\}.$$

Definīcija Galīga automāta ar izeju A BC-sarežģītība $C_{\text{BC}}(A)$ ir minimālā sarežģītība, kāda tam ir pie kāda kodējuma:

$$C_{\text{BC}}(A) = \min\{C_{\text{BC}}(A, E(A)) : E(A) \text{ ir GDA } A \text{ kodējums}\}.$$

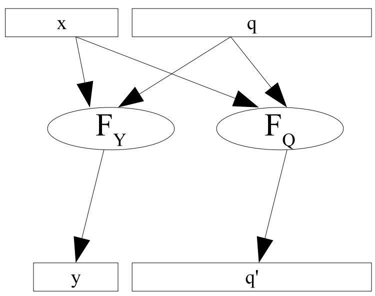
Nākamā teorēma ir 4.7 teorēmas analogs GDA ar izeju. Rezultāti arī ir ļoti līdzīgi, neliela atšķirība ir tikai gadījumā, kad $|X| = 1$ un $|Y| > 2$.

8.2. Teorēma *Ja $|X| = k \geq 2$ tad jebkuram galīgam automātam ar izeju A ar s stāvokļiem,*

$$\lceil \log s \rceil \leq C_{\text{BC}}(A) \lesssim (k-1)s.$$

Ja $|X| = 1$ tad jebkuram galīgam automātam ar izeju A ar s stāvokļiem,

$$\lceil \log s \rceil \leq C_{\text{BC}}(A) \lesssim \log |Y| \frac{s}{\log s}.$$



8.2. att. Pārejas shēmas optimāla konstruēšana automātam ar izeju

Pierādījums Novērtējumu no apakšas pierāda tieši tāpat ka akceptoriem. Novērtējumu no augšas pierāda ļoti līdzīgi.

Konstruēsim automāta loģiskās shēmas reprezentāciju, kur tā pārejas funkcija F sastāvēs no divām atsevišķām shēmām: nākamajam stāvoklim F_Q un izejas simbolam F_Y (8.2. zīm.). Šādā gadījumā funkcija F_Y spēlēs to pašu lomu, ko akceptēšanas shēma G GDA bez izejas. Tās sarežģītību var novērtēt, izmantojot 2.4. teorēmu:

$$C(F_y) \lesssim \frac{\log|Y| \cdot s}{\log\log|Y| + \log s} \lesssim \frac{\log|Y| \cdot s}{\log s}.$$

(šis novērtējums gan formāli ir spēkā tikai tad, ja $|Y|$ ir divnieka pakāpe, bet, izmantojot 2.4. teorēmas vispārinājumu no [28], kur šī teorēma aplūkota arī gadījumā, kad dažas no izejas vērtībām netiek pieņemtas nevienai argumenta vērtībai, to var pierādīt arī patvaļīgiem $|Y|$.)

Ja $k \geq 2$, tad dominējošais svars BC-sarežģītībā ir loģiskajai shēmai F_Q, kuras sarežģītību var novērtēt tieši tāpat, kā to dara GDA bez izejas, iegūstot, ka $C_{\text{BC}}(A) \lesssim C(F_Q) \lesssim (k-1)s$.

Bet, ja $k = 1$, tad dominējošā sarežģītība ir loģiskajai shēmai F_Y, tāpēc $C_{\text{BC}}(A) \lesssim C(F_Y) \lesssim \frac{\log|Y| \cdot s}{\log s}$. ∎

Abus piemērus, kas seko 4.7. teorēmai (4.4. nodaļa), un parāda, ka šīs robežas ir vairāk vai mazāk sasniedzamas, arī ir viegli pielāgot galīgiem automātiem ar izeju. Aplūkotajiem automātiem, kas pazīst valodas L_n un L_n^{Sh} var konstruēt atbilstošus automātus ar izeju, kuru stāvokļu pārejas funkcijas sakrīt, bet izejas funkcija sakrīt ar harakterisko funkciju akceptora beigu stāvokļu kopai. Tas noved pie tā, ka arī galīgiem automātiem ar izeju ir sasniedzama 8.2 teorēmas apakšējā robeža BC-sarežģītībai $\lceil \log s \rceil$, un var atrast piemēru valodai, ar ne vairāk kā s stāvokļiem, bet BC-sarežģītību lielāku par $s/(\log s)^2$.

Galīgiem automātiem ar izeju ir spēkā analoģiski BC-sarežģītības apakšējie novērtējumi gandrīz visiem automātiem. Tie ir gandrīz identiski vairāku simbolu ieejas alfabēta gadījumā, bet nedaudz atšķiras viena burta ieejas alfabēta gadījumā.

8.3. Teorēma *Fiksēsim ieejas alfabētu X un izejas alfabētu Y un ar $\mathfrak{B}_s$ apzīmēsim visu minimālo automātu kopu ar ne vairāk ka s stāvokļiem ar šādiem ieejas un izejas alfabētiem. Tad gandrīz visien $B \in \mathfrak{B}_s$*

$$C_{\text{BC}}(B) \gtrsim (k-1)s, \qquad \textit{ja } |X| = k \geq 2,$$

$$C_{\text{BC}}(B) \gtrsim \log|Y| \frac{s}{\log s}, \qquad \textit{ja } |X| = 1.$$

Pierādījums Gadījumā, kad $k \geq 2$, dažādu minimālo automātu skaitu (no 8.1. teorēmas) var novērtēt analogi kā iepriekš un arī tālākais pierādījums ir tieši tāds pats kā 4.10. teorēmā.

Ja $k = 1$, tad no 8.1. teorēmas izriet, ka $|\mathfrak{B}_s| \geq |Y|^s$. Tad

$$\frac{\log \mathfrak{B}_s}{\log\log \mathfrak{B}_s} \geq \frac{\log|Y|^s}{\log\log|Y|^s} = \frac{s\log|Y|}{\log s + \log\log|Y|} \gtrsim \log|Y| \frac{s}{\log s}. \quad ∎$$

Tādējādi Šenona efekts ir spēkā arī automātiem ar izeju:

8.4. Sekas *Fiksēsim ieejas alfabētu X un izejas alfabētu Y un ar $\mathfrak{B}_s$ apzīmēsim visu minimālo automātu kopu ar ne vairāk ka s stāvokļiem ar šādiem ieejas un izejas alfabētiem. Tad gandrīz visien $B \in \mathfrak{B}_s$*

$$(k-1)s \lesssim C_{\text{BC}}(B) \lesssim (k-1)s, \qquad \textit{ja } |X| = k \geq 2,$$

$$\frac{\log|Y| \cdot s}{\log s} \lesssim C_{\text{BC}}(B) \lesssim \frac{\log|Y| \cdot s}{\log s}, \qquad \textit{ja } |X| = 1.$$

Kā pēdējo šajā nodaļā par galīgiem automātiem ar izeju aplūkosim to BC-sarežģītības saistību ar stāvokļu minimizāciju. Arī automātiem ar izeju var pierādīt 7.1. teorēmai analogu teorēmu: minimizējot automātu, tā BC-sarežģītība var pieaugt vairāk nekā polinomiāli.

8.5. Teorēma *Ja eksistē tāds polinoms $p(x)$, ka visiem galīgiem automātiem ar izeju B binārā alfabētā izpildās $C_{\text{BC}}(M(B)) < p(C_{\text{BC}}(B))$, tad PSPACE$\subseteq$ P/poly.*

Pierādījums GDA A_n^V vietā ņemsim galīgu automātu B_n^V ar bināru ieeju un izeju, kura stāvokļu pārejas funkcija (un pārejas shēma) sakrīt ar A_n^V pārejas funkciju, bet izejas funkcijas (shēma) sakrīt ar B_n^V akceptēšanas shēmu. Automāts B_n^V drukās uz lentes vieninieku tikai $2^m + n$ pozīcijā un tikai tad, ja $w_1 w_2 \ldots w_n \in V$. Pārējais pierādījums ir tieši tāds pats kā 7.1. teorēmā. ∎

9. nodaļa

Algoritmiskā sarežģītība GDA loģiskās shēmas reprezentācijā

Līdz šim mēs aplūkojām tikai GDA "aprakstošo" sarežģītību, cik sarežģīti ir doto GDA aprakstīt kā loģisko elementu shēmu. Bet iesim tālāk un pamēģināsim noskaidrot, cik sarežģīti ir veikt dažādas operācijas ar GDA to loģiskās shēmas reprezentācijā.

GDA standarta reprezentācijai šādi jautājumi nav pārāk interesanti, jo šo operāciju sarežģītība ir neliela. Ja aplūkojam tādus jautājumus kā stāvokļu sasniedzamība, stāvokļu ekvivalence, GDA ekvivalence vai to minimizācija, tad tie visi ir izrēķināmi polinomiālā laikā, daži no tiem (stāvokļu sasniedzamība) pieder pat klasei NL un ir NL pilni attiecībā pret rudimentāru redukciju[20]. Šīs vienkāršības dēļ tie netiek pārāk daudz aplūkoti literatūrā, izņemot GDA minimizāciju, kurš ir praktiski nozīmīgs algoritms un ne tik vienkāršs kā pārējie.

Bet situācija kardināli mainās, ja mēs šos jautājumus aplūkojam GDA, kas doti to loģiskās shēmas reprezentācijā. Izrādās, ka tādā gadījumā visi šie iepriekšminētie jautājumi ir PSPACE-pilnīgi.

Mēs sāksim ar to, ka parādīsim, ka stāvokļu sasniedzamības uzdevums GDA, kas doti loģiskās shēmas reprezentācijā, ir PSPACE-pilnīgs . Lai to izdarītu, mēs lietosim teoriju par algoritmisku instanču īsajām reprezentācijām (succinct representations of algorithmic instances), kas tika attīstīta pagājušā gadsimta 80. un 90. gados[27][4][7]. Katrs nākamais raksts šajā sērijā vispārina iepriekšējo, bet mazliet maina arī definīcijas, tāpēc mēs izmantosim pēdējo no tiem[7].

Aplūkosim loģisko elementu shēmu $c(x_1, \ldots, x_n)$ ar n ieejas mainīgajiem un vienu izejas mainīgo, tā dabiskā veidā apraksta vārdu $w(c)$ binārā alfabētā garumā 2^n: tā i-tais simbols ir c vērtība i-tajai ieejas mainīgo vērtībai, kur tās ir sakārtotas leksikogrāfiski. Citiem vārdiem sakot, $w(c)$ ir izejas kolonna loģiskās shēmas c patiesumvērtību tabulai.

Šādā veidā jebkuru vārdu, kura garums ir divnieka pakāpe, var aprakstīt (bezgalīgi daudzos veidos) ar loģisko shēmu. Bet, lai tā varētu aprakstīt arī visus pārējos vārdus, ieviesīsim vēl vienu apzīmējumu.

Ar $w(c, m)$ apzīmēsim vārdu, ko veido vārda $w(c)$ pirmie m simboli (pārējie tiek atmesti). Ievērosim, ka tas nozīmē, ka $w(c) = w(c, 2^n)$, kur n ir c ieejas mainīgo skaits. Šeit mēs pieņemsim, ka loģiskā shēma c ir kaut kā iekodēta binārā formā. Ja x nav nevienas loģiskās shēmas kodējums, tad pieņemsim, ka $w(x, m)$ apzīmē tukšo vārdu. Mēs teiksim, ka pāris (c, m) ir vārda $w(c, m)$ *īsā reprezentācija*. Valodai L ar $S(L)$ apzīmēsim visu L vārdu visu īso reprezentāciju kopu:

$$S(L) = \{(c, m) : w(c, m) \in L\}.$$

Kāpēc tad (c, m) sauc par vārda $w(c, m)$ īso reprezentāciju? Lielākajai daļai vārdu īsā reprezentācija nemaz nebūs īsāka par pašu vārdu, taču, ja vārdam ir kāda vienkārša struktūra, tad iespējams, ka to var aprakstīt pat ar eksponenciāli īsāku loģisko shēmu nekā vārda garums. Piemēram, vārda, kas sastāv no $2^n - 1$ nulles un viena vieninieka, īsā reprezentācija var būt $(c_{\&}, 2^n)$, kur $c_{\&}$ ir loģiskā shēma ar n ieejas mainīgajiem, kas atgriež visu savu ieejas mainīgo konjunkciju.

Šeit un turpmāk par īsās reprezentācijas izmēru sauksim tās garumu bitos un apzīmēsim ar $|(c, m)| = |c| + |m|$, kur $|m| = \lceil \log m \rceil$ ir skaitļa m garums bitos, bet $|c|$ ir loģiskās shēmas c binārā kodējuma garums bitos, kas nav tas pats, kas tās loģisko elementu skaits, bet ir ar to cieši (polinomiāli) saistīts. Loģisko shēmu ar n ieejām un l loģiskajiem elementiem var iekodēt $l(2 \lceil \log (l + n) \rceil + 2)$ bitos: visas ieejas un loģiskos elementus var iekodēt $\lceil \log (l + n) \rceil$ bitos un katram loģiskajam elementam jānorāda tā tips ($\&, \vee, \neq$, pietiek ar 2 bitiem) un divi ieejas argumenti ($2 \lceil \log (l + n) \rceil$ biti).

Parasti, lai pamatotu, ka jautājums ir pilnīgs kādai algoritmiskās sarežģītības klasei, pietiek ar polinomiāla laika redukciju, vai sliktākajā gadījumā logaritmiskas lentes sarežģītības redukciju (to lieto, lai sīkāk pētītu polinomiālā laikā izrēķināmas funkcijas, piemēram, klasi NL). Mums būs vajadzīgs vēl smalkāks redukcijas rīks — polilogaritmiska laika redukcija $\leq^{PLT}$.

Tjūringa mašīna, kas strādā polilogaritmiskā laikā, ir stipri ierobežota, tā nevar pat nolasīt visu savu ieejas lenti. Lai to pārvarētu, tiks lietota jau zināma Tjūringa mašīnas modifikācija — brīvpiekļuves (random access) Tjūringa mašīna, kas vienā solī var nolasīt jebkuru vienu bitu no savas ieejas lentes.

Papildus ieejas lentei un darba lentei, tai būs vēl divas speciālas lentes: lente, uz kuras uzrakstīt indeksu (kārtas numuru) kādam ieejas bitam, un lente viena bita garumā, uz kuras šis ieejas bits tiks uzrakstīts. Šai Tjūringa mašīnai būs arī īpašs "lasīt" stāvoklis, uz kuru pārejot, tā vienā solī nolasa attiecīgo ieejas bitu (kura indekss uzrakstīts uz pirmās papildu lentes), kas tad parādās uz otrās papildu lentes.

Otra problēma ar polilogaritmiska laika Tjūringa mašīnām ir, ka tās rēķinātās funkcijas vērtības garums var būt maksimums polilogaritmisks attiecībā pret ieejas garumu. Lai tiktu galā ar šo problēmu, prasīsim, nevis lai viss rezultāts ir izrēķināms polilogaritmiskā laikā, bet jebkurš rezultāta izejas bits ir izrēķināms polilogaritmiskā laikā.

Definīcija Valoda L ir polilogaritmiskā laikā reducējama uz valodu L' ($L \leq^{PLT} L'$), ja var atrast divas funkcijas $R : \Sigma^* \times \mathbb{N} \to \{0, 1\}$ un $l : \Sigma^* \to \mathbb{N}$, kas ir izrēķināmas polilogaritmiskā laikā ar brīvpiekļuves Tjūringa mašīnu un kam izpildās īpašība

$$x \in L \leftrightarrow R(x, 0)R(x, 1) \dots R(x, l(x) - 1) \in L'.$$

Klasiskajā polinomiālā laika redukcijā tiek prasīts, lai būtu tāda polinomiālā laikā izrēķināma funkcija f, kurai $x \in L \leftrightarrow f(x) \in L'$. Mūsu gadījumā šādas funkcijas f nav, bet ir divas citas funkcijas: $l(x)$, kas izrēķina $f(x)$ garumu un $R(x, i)$, kas izrēķina $f(x)$ i-to bitu.

Polilogaritmiska laika redukcija ir refleksīva un tranzitīva un no tās izriet polinomiāla laika redukcija.

Teiksim, ka valoda L' ir $\leq^{PLT}$ grūta kādai valodu klasei $\mathfrak{L}$, ja jebkura valoda $L \in \mathfrak{L}$ ir polilogaritmiskā laikā reducējama uz L'. Ar DTIME($f(n)$) (DSPACE($f(n)$)) apzīmēsim valodu klasi, ko ar determinētu Tjūringa mašīnu var izrēķināt laikā (ar lentes sarežģītību) $f(n)$, attiecīgi ar NTIME($f(n)$) (NSPACE($f(n)$)) apzīmēsim valodu klasi, ko var izrēķināt laikā (ar lentes sarežģītību) $f(n)$ ar nedeterminētu Tjūringa mašīnu.

Galvenais rezultāts no [7], ko mēs izmantosim, ir sekojošs:

9.1. Teorēma *[7] Ja kādai nedilstošai funkcijai $f(n)$ valoda L ir $\leq^{PLT}$ grūta valodu klasei DTIME($f(n)$), NTIME($f(n)$), DSPACE($f(n)$) vai NSPACE($f(n)$), tad $S(L)$ ir $\leq^{PLT}$ grūta attiecīgi valodu klasei DTIME($f(2^n)$), NTIME($f(2^n)$), DSPACE($f(2^n)$) vai NSPACE($f(2^n)$).*

No šīs teorēmas mēs gan izmantosim tikai to daļu, kas attiecas uz sarežģītības klasēm DSPACE($f(n)$) un NSPACE($f(n)$), bet pārējās klases tajā ir atstātas pilnīguma pēc un, lai saglabātu atbilstību oriģinālam.

Tagad atgriezīsimies pie galīgiem automātiem un paskatīsimies, kā tie ir saistīti ar visu šo teoriju. GDA loģiskās shēmas reprezentācija (F, G) un tā īsā reprezentācija (c, m) ir savā ziņā saistītas lietas, abas ar loģiskajām shēmām apraksta GDA struktūru. Arī izmēra ziņā viena no otras neatšķiras vairāk kā polinomiāli un arī viena no otras ir iegūstama polinomiālā laikā.

Intuitīvi tam var viegli noticēt, bet, lai to precīzi pamatotu, vispirms noskaidrosim, kā tieši izskatās GDA īsā reprezentācija, un tam savukārt mums vajadzēs noskaidrot, kā izskatās GDA, kas ir kodēts par bināru virkni. Ņemsim kādu minimālu stāvokļu kodējumu f_Q un minimālu ieejas kodējumu f_Σ un aprakstīsim GDA ar bināru virkni garumā $|\Sigma| \cdot |Q| \cdot b_Q + |Q|$. Pirmie $|\Sigma| \cdot |Q| \cdot b_Q$ tās biti apraksts stāvokļu pārejas tabulu, tajos visām $|\Sigma| \cdot |Q|$ iespējamajām ieejas vērtībām (ieejas simbols + stāvoklis) pēc kārtas būs uzrakstīts nākamā stāvokļa kodējums (garumā b_Q). Pēdējie Q biti noteiks, kuri stāvokļi ir akceptējoši, katram stāvoklim atbildīs viens bits, atkarībā no kura vērtības (1 vai 0) stāvoklis būs attiecīgi akceptējošs vai noraidošs. Sākuma stāvoklis tiek kodēts ar visām nullēm.

Tādējādi GDA ar $|Q|$ stāvokļiem alfabētā Σ īsā reprezentācija būs (c, m), kur $m = |\Sigma| \cdot |Q| \cdot \lceil \log Q \rceil + |Q|$, bet c ir loģiskā shēma ar $\lceil \log m \rceil$ ieejām un vienu izeju, kas uz savām pirmajām m ieejām atgriež attiecīgos GDA binārā kodējuma bitus.

Atgriežoties pie sākotnējā jautājuma par GDA loģiskās shēmas reprezentācijas un īsās reprezentācijas saistību, tagad varam redzēt, ka pirmā ar divām loģiskajām shēmām apraksta attiecīgi GDA pārejas funkciju un akceptējošo stāvokļu kopu, bet otrā ar vienu loģisko shēmu bitu pa bitam aprasta vispirms vienu un pēc tam otru.

Pirms ķerties pie šīs saistības formāla apraksta, vēl jāpiezīmē, ka šī saistība starp GDA loģiskās shēmas reprezentāciju un īso reprezentāciju ir kaut kādā ziņā neviennozīmīga. Dota loģiskās shēmas reprezentācija (F, G) pat pie fiksēta ieejas alfabēta kodējuma var atbilst vairākiem GDA (kas atšķiras ar nesasniedzamiem stāvokļiem), par šo skat. 3. nodaļu. Šajā gadījumā mēs par "īsto" uzskatīsim GDA ar maksimālo stāvokļu skaitu 2^{b_Q}. Tiesa, nesasniedzamo stāvokļu izmešana pēc tam var būt sarežģīts uzdevums.

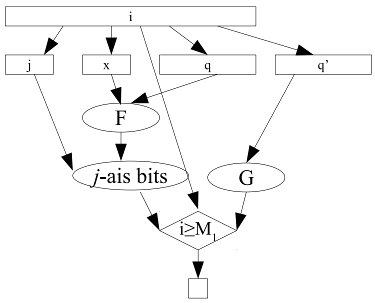
9.2. Teorēma *Fiksēsim ieejas alfabētu Σ un tā kodējumu f_Σ. Ja dota GDA loģiskās shēmas reprezentācija (F, G), tad no tās polinomiālā laikā iespējams iegūt īso rep-*

rezentāciju (c, m) kādam GDA, kas no dotā, iespējams, atšķiras tikai ar kādiem nesasniedzamiem stāvokļiem, pie tam $|(c, m)| < poly(C_{\text{BC}}((F, G)))$.

Ja dota GDA īsā reprezentācija (c, m), tad no tās polinomiālā laikā iespējams iegūt šī paša GDA loģiskās shēmas reprezentāciju (F, G), pie tam $C_{\text{BC}}(F, G) < poly(|(c, m)|)$.

Pierādījums Pieņemsim, ka mums ir dota GDA loģiskās shēmas reprezentācija (F, G) pie kodējuma (f_Σ, f_Q), attiecīgi, garumā b_Σ un b_Q, pie tam ieejas kodējums f_Σ ir minimāls, un mums nepieciešams uzkonstruēt tā īso reprezentāciju (c, m). Šī reprezentācija var atbilst vairākiem GDA (gadījumā, ja kādi stāvokļi ir nesasniedzami), mēs konstruēsim maksimālo iespējamo GDA ar 2^{b_q} stāvokļiem.

GDA binārā kodējuma garums būs $m = |\Sigma| \cdot 2^{b_Q} \cdot b_Q + 2^{b_Q}$. Loģiskajai shēmai c, saņemot ieejā indeksu i, jāatgriež stāvokļa bits kādam nākamajam stāvoklim, ja $0 \leq i < M = |\Sigma| \cdot 2^{b_Q} \cdot b_Q$, vai $(i - M)$-ā stāvokļa "akceptēšanas bits", ja $i \geq M$. Shematiski tas, kā no loģiskajām shēmām F un G iegūt loģisko shēmu c, attēlots 9.1 zīmējumā.



9.1. att. Īsās reprezentācijas loģiskās shēmas c konstrukcija no GDA loģiskās shēmas reprezentācijas (F, G).

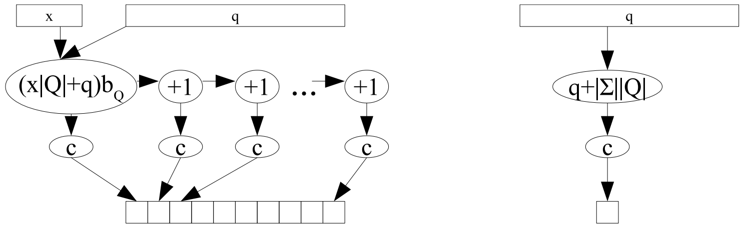
Vispirms no dotā indeksa i jāizrēķina atbilstošais ieejas simbols x, ieejas stāvoklis q un izejas stāvokļa bits j, gadījumā, ja i norāda uz kādu bitu stāvokļu pārejas tabulā, kā arī stāvoklis q', gadījumā, ja i norāda uz kādu bitu akceptēšanas tabulā. Šie aprēķini, lai nesarežģītu loģisko shēmu, tajā nav parādīti, bet ir diezgan vienkārši, ar (x, q) apzīmēta šo abu argumentu bināro vērtību konkatenācija.

$$
\begin{aligned}
j &= i \mod b_Q, \\
(x, q) &= \frac{i - j}{b_Q}, \\
q' &= i - M.
\end{aligned}
$$

Tālāk ar pārejas shēmu F jāizrēķina nākamais stāvoklis, no kura jāizvēlas atbilstošais bits, ja $i < M$, vai arī jāatgriež akceptēšanas shēmas izrēķinātais bits, ja $i \geq M$.

Redzams, ka loģiskās shēmas c loģisko elementu skaits ir polinomiāls attiecībā pret b_Q, $C(F)$ un $C(G)$ (jo i garums ir polinomiāls attiecībā pret b_Q), tātad tas ir arī polinomiāls attiecībā pret $C_{\text{BC}}((F, G))$. Tātad arī $|(c, m)|$ garums bitos ir polinomiāli ierobežots ar $C_{\text{BC}}((F, G))$.

Aplūkosim tagad šo pārveidojumu pretējā virzienā. Pieņemsim, ka mums ir dota kāda GDA īsā reprezentācija (c, m) un mums nepieciešams konstruēt tā loģiskās shēmas reprezentāciju (F, G). Loģisko shēmu F iespējams konstruēt, apvienojot



9.2. att. GDA loģiskās shēmas reprezentācijas konstrukcija no tā īsās reprezentācijas

paralēli b_Q loģiskās shēmas c, katru vienam izejas bitam. Savukārt, loģiskajai shēmai G pietiek ar vienu loģisko shēmu c un pareizi tai padotu parametru (9.2. zīm.). Redzams, ka $C_{\text{BC}}((F,G))$, šajā gadījumā ir polinomiāli atkarīga $C(c)$, b_Q un b_Σ, tātad tā ir ne vairāk kā polinomiāli atkarīga no (c,m) garuma bitos. ∎

Nākamā teorēma ir diezgan līdzīga klasiskajai teorēmai par to, ka orientēta grafa virsotņu sasniedzamība ir NL-pilnīgs uzdevums. Bet tā kā mums nepieciešams, lai šī NL-pilnība būtu attiecībā pret $\leq^{PLT}$, redukciju, tad pierādīsim šo teorēmu pilnībā. Atgādināsim, ka ar L un NL apzīmē valodu klases, kas ir izrēķināmas, lietojot tikai logaritmisku lentes garumu, ar attiecīgi determinētu vai nedeterminētu Tjūringa mašīnu.

Ar REACH apzīmēsim valodu, kas sastāv no visiem pāriem (s,A), kur s ir kāds GDA A sasniedzams stāvoklis. Šeit A ir dots tā standarta (binārajā) reprezentācijā.

9.3. Teorēma *Fiksētam ieejas alfabētam $|\Sigma| \geq 2$, REACH ir NL-pilnīga attiecībā pret $\leq^{PLT}$ redukciju.*

Unāram ieejas alfabētam REACH ir L-pilnīga attiecībā pret $\leq^{PLT}$ redukciju.

Pierādījums Sāksim ar gadījumu, kad $|\Sigma| \geq 2$, un vispirms parādīsim, ka REACH $\in$ NL. Dotam GDA A mēs varam nedeterminēti pa vienam simbolam uzminēt to ieejas virkni, ar kuru no sākuma stāvokļa var nokļūt dotajā stāvoklī s. Lai to izdarītu, uz darba lentes mums jāglabā tikai tekošais stāvoklis un nākamais stāvoklis, kas kopā aizņem ne vairāk kā logaritmiski daudz lentes attiecībā pret ieejas lentes garumu.

Tālāk pierādīsim, ka jebkura valoda $B \in$ NL ir $\leq^{PLT}$ reducējama uz REACH. Ņemsim nedeterminētu Tjūringa mašīnu M binārā alfabētā, ar atsevišķu darba lenti, kas pazīst B un tam netērē vairāk par $c \log n$ darba lentes rūtiņām. Katrai tās ieejai x konstruēsim GDA A_x un tā stāvokli s_x, kurš būs sasniedzams tad un tikai tad, ja M akceptē x.

Aplūkosim M konfigurāciju grafu V_x, kas veidojas, tai darbojoties uz ieejas vērtības x. Tā kā tā ir nedeterminēta Tjūringa mašīna, tad no katras V_x virsotnes iziet divas (nedeterminēta pāreja), viena (determinēta pāreja) vai 0 (beigu stāvoklis) šķautnes. GDA A_x stāvokļi būs V_x virsotnes (M konfigurācijas), tā pārejas būs V_x šķautnes. Tā kā alfabētā Σ ir vismaz divi simboli, tad mēs varam katra stāvokļa izejošajām šķautnēm piekārtot dažādus simbolus un, lai A būtu pilnībā definēts, varam uzskatīt, ka visas pārējās šķautnes (ar neizmantotajiem simboliem) paliek tajā pašā stāvoklī.

GDA A_x sākuma stāvoklim atbilst konfigurācija, kurai uz ieejas lentes ir rakstīts x, darba lente ir tukša, galviņas atrodas abu lenšu kreisajā malā un M stāvoklis ir tās sākuma stāvoklis. Stāvoklim s_x, kura sasniedzamība mūs interesē, atbilst konfigurācijai, kurā M atrodas tās beigu stāvoklī, galviņas atrodas lenšu kreisajā malā un uz tās darba lentes rakstīts "1" (tā akceptē vārdu x). Redzams, ka šis stāvoklis būs sasniedzams tad un tikai tad, ja M akceptē x.

Atliek pamatot, ka A_x izmērs ir polinomiāls attiecībā pret $|x|$ un šī redukcija ir $\leq^{PLT}$ redukcija. Ja M izmanto ne vairāk, kā $c \log n$ no tās darba lentes, tad darba lentei ir iespējami maksimums $3^{c \log |x|}$ stāvokļu (katrs simbols uz darba lentes ir 0, 1 vai λ), ieejas lentes galviņai ir $|x|$ iespējami stāvokļi, darba lentes galviņai — $c \log |x|$ stāvokļi un vēl pašai M iespējami Q_M stāvokļi. Tātad kopējais iespējamo konfigurāciju skaits ir:

$$K_M = 3^{c \log |x|} \cdot |x| \cdot c \log |x| \cdot Q_M,$$

kas ir polinomiāls attiecībā pret $|x|$.

Lai pamatotu, ka šo redukciju var izveikt ar brīvpieejas Tjūringa mašīnu polilogaritmiskā laikā, vispirms ievērosim, ka polilogaritmiskā laikā iespējams aprēķināt ieejas datu garumu $|x|$. To var izdarīt, piemeklējot $|x|$ ar binārās meklēšanas metodi. Sākot ar vērtību 1, katrā solī var palielināt indeksa vērtību divas reizes, līdz beidzot tas ir garāks par doto vārdu, un tad ar intervāla dalīšanas metodi atrast precīzo $|x|$ vērtību.

Otrkārt, ja mums ir dota $|x|$ vērtība (kuras garums ir $\log |x|$), tad visus algebriskos aprēķinus, kas ar to jāveic: (saskaitīšanu, reizināšanu, kāpināšanu, logaritmēšanu), kuru rezultāta garums ir polinomiāls attiecībā pret $\log |x|$, var izdarīt polinomiālā laikā attiecībā pret $\log |x|$, kas ir polilogaritmiskā laikā attiecībā pret $|x|$. Tas nozīmē, ka K_M vērtību (kas ir polinomiāla attiecībā pret $|x|$) var izrēķināt polilogaritmiskā laikā attiecībā pret $|x|$.

Polilogaritmiskajai redukcijai mums vispirms jāspēj izrēķināt iegūtā GDA A_x garums bitos $l(x)$. Tam ir K_M stāvokļu, no katra iziet $|\Sigma|$ pārejas, tātad tā stāvokļu pārejas tabulas izmērs ir $K_M \cdot |\Sigma| \cdot \lceil \log K_M \rceil$, bet akceptēšanas tabulas izmērs ir K_M. Tātad kopējais A_x izmērs bitos ir

$$l(x) = K_M \cdot |\Sigma| \cdot \lceil \log K_M \rceil + K_M,$$

ko var izrēķināt polinomiālā laikā attiecībā pret K_M izmēru, tātad polilogaritmiskā laikā attiecībā pret $|x|$.

Otrkārt mums ir jāspēj izrēķināt jebkurš bits no A_x pieraksta $R(x, i)$. Ja $i < K_M \cdot |\Sigma| \cdot \lceil \log K_M \rceil$, tad tas ir bits no stāvokļa pāreju tabulas un to mēs varam iegūt no M stāvokļa pāreju tabulas un atbilstošās sākuma konfigurācijas, (ko var iegūt no i). Šīs konfigurācijas izmērs ir logaritmisks attiecībā pret $|x|$, tāpēc arī tai sekojošo konfigurāciju (kurā ir pamainījos ieejas lente un iespējams pabīdījušās galviņas) vai izrēķināt polilogaritmiskā laikā attiecībā pret $|x|$. Ja $i \geq K_M \cdot |\Sigma| \cdot \lceil \log K_M \rceil$, tad tas ir bits no akceptējošo stāvokļu tabulas un tas būs vieninieks tad un tikai tad, ja atbilstošā konfigurācija atbilst M beigu stāvoklim, kurš akceptē vārdu x. Šajā beigu stāvoklī uz darba lentes ir rakstīts "1", galviņas atrodas sākuma stāvoklī, bet M — beigu stāvoklī, lai to pārbaudītu, pietiek ar polilogaritmisku (attiecībā pret $|x|$) laiku.

Unāram alfabētam pierādījums ir tāds pats, atliek tikai ievērot, ka, tā kā šajā gadījumā mums Tjūringa mašīna būs determinēta, tad no katras tās konfigurāciju grafa virsotnes būs maksimums viena izejošā pāreja, kurai tad arī var piekārtot vienīgo alfabēta simbolu. ∎

Apzīmēsim ar REACH$_{LSR}$ valodu, kas sastāv no visiem pāriem $(s,(F,G))$, kur (F,G) ir kāda GDA loģiskās shēmas reprezentācija, bet s ir kāda šī GDA sasniedzama stāvokļa kodējums (LSR ir saīsinājums no Loģiskās Shēmas Reprezentācijas).

9.4. Teorēma *Fiksētam ieejas alfabētam REACH$_{LSR}$ ir PSPACE-pilnīga valoda.*

Pierādījums Vispirms parādīsim, ka REACH$_{LSR}$ atrodas klasē PSPACE t.i. to var atrisināt, izmantojot polinomiālu darba lentes garumu. Līdzīgi kā ar REACH mēs varam nedeterminēti simbolu pēc simbola uzminēt ieeju, kura noved šo GDA vajadzīgajā stāvoklī un to pārbaudīt Tam mums nav vajadzīgs vairāk kā polinomiāls lentes daudzums, tāpēc REACH$_{LSR}$ $\in$NPSPACE. Bet tā kā NPSPACE = PSPACE (Saviča teorēma), tad REACH$_{LSR}$ $\in$ PSPACE.

No 9.3 un 9.1 teorēmām tiešā veidā izriet, ka $S(REACH)$ ir PSPACE-grūta valoda. Bet S(REACH) var polinomiālā laikā reducēt uz REACH$_{LSR}$ (9.2 teorēma), tāpēc arī REACH$_{LSR}$ ir PSPACE-grūta. ▮

Izmantojot šo kā bāzi, mēs varam pierādīt, ka vēl arī citi klasiski jautājumi par GDA īpašībām ir PSPACE-pilnīgi, ja GDA mums ir uzdots loģiskās shēmas reprezentācijā.

9.5. Teorēma *Sekojoši jautājumi ir PSPACE-pilnīgi:*

1. *Ja dota GDA loģiskās shēmas reprezentācija un divu tā stāvokļu kodējums, noskaidrot, vai šie stāvokļi ir ekvivalenti.*
2. *Ja dota GDA loģiskās shēmas reprezentācija, noskaidrot, vai valoda, ko tas akceptē, ir tukša.*
3. *Ja dotas divu GDA loģiskās shēmas reprezentācijas, noskaidrot, vai tie ir ekvivalenti.*
4. *Ja dota GDA loģiskās shēmas reprezentācija un skaitlis k, noskaidrot, vai eksistē tam ekvivalents GDA, kura BC-sarežģītība nepārsniedz k.*

Pierādījums Katram no šiem jautājumiem vispirms parādīsim, ka tie pieder klasei PSPACE, un pēc tam — ka tie ir PSPACE-grūti.

1. Pieņemsim, ka divi dotie stāvokļi nav ekvivalenti. Tas nozīmē, ka eksistē kāda ieejas simbolu virkni, kuru, sākot lasīt pirmajā stāvoklī, GDA nonāks kādā akceptējošā stāvoklī, bet, sākot lasīt otrajā stāvoklī — kādā noraidošā (vai otrādi). Mēs varam nedeterminēti (simbolu pēc simbola) uzminēt šo ieeju un tad pārbaudīt, ka tā tiešām noved divus dotos stāvokļus pie diviem stāvokļiem, no kuriem viens ir akceptējošs, bet otrs noraidošs. No tā varam secināt, ka stāvokļu neekvivalences pārbaude pieder klasei NPSPACE, tātad arī PSPACE. Bet tad arī stāvokļu ekvivalences pārbaude pieder klasei PSPACE.

   Tagad parādīsim kā REACH$_{LSR}$var reducēt uz stāvokļu ekvivalences pārbaudi. Pieņemsim, ka mums ir dota GDA loģiskās shēmas reprezentācija (F,G) un kāda stāvokļa kodējums s, kura sasniedzamību mēs gribam pārbaudīt. Pārveidosim doto loģiskās shēmas reprezentāciju, atstājot s par vienīgo akceptējošo stāvokli un pieliekot klāt vienu nesasniedzamu stāvokli s', no kura visas pārejas pāriet uz viņu pašu. Redzams, ka pārveidotājā GDA sākuma stāvoklis un s' būs ekvivalenti tad un tikai tad, ja sākotnējā GDA stāvoklis s nebija sasniedzams.

2. Lai noskaidrotu, vai dotā GDA atpazītā valoda nav tukša, mēs varam, līdzīgi kā pirmajam jautājumam, nedeterminēti, simbolu pēc simbola, uzminēt ieeju, kas novedīs to kādā akceptējošā stāvoklī, tas nozīmē, ka šis jautājums pieder klasei NPSPACE = PSPACE. Lai parādītu, ka šis jautājums ir PSPACE-pilnīgs, atkal pieņemsim, ka mums ir dota loģiskās shēmas reprezentācija (F, G) un stāvokļa kodējums s, kura sasniedzamību mēs gribam pārbaudīt. Pārveidosim akceptēšanas shēmu G tā, lai s būtu vienīgais akceptējošais stāvoklis. Acīmredzami, ka šī jaunā GDA atpazītā valoda būs tukša tad un tikai tad, ja s nav sasniedzams.

3. Lai parādītu, ka trešais jautājums pieder klasei PSPACE, mēs tāpat kā pirmajam jautājumam varam nedeterminēti pārbaudīt abu šo GDA sākuma stāvokļu neekvivalenci. Lai parādītu, ka šis jautājums ir PSPACE-pilnīgs, mēs varam ievērot, ka pat pārbaudīt vai dotā GDA loģiskās shēmas reprezentācija ir ekvivalenta neko neakceptējoša GDA (kas pazīst tukšo valodu) loģiskās shēmas reprezentācijai, ir PSPACE-grūts uzdevums (otrais jautājums).

4. Mēs varam nedeterminēti uzminēt dotajam GDA ekvivalenta GDA loģiskās shēmas reprezentāciju ar k vai mazāk stāvokļiem un tad pārbaudīt to šo abu GDA ekvivalenci. Tātad šis jautājums pieder klasei NPSPACE = PSPACE. Lai parādītu, ka tas ir PSPACE-pilnīgs, reducēsim uz to jautājumu par to, vai GDA atpazītā valoda ir tukša (2. jautājums). Ievērosim, ka vienīgās divas GDA loģiskās shēmas reprezentācijas, kuru sarežģītība ir 0, ir visu akceptējoša un visu noraidoša GDA loģiskās shēmas reprezentācijas, tiem vienīgajiem nav neviena stāvokļa bita. Pieņemsim, ka mums dota GDA loģiskās shēmas reprezentācija un mums jānoskaidro, vai tā atpazītā valoda ir tukša. Vispirms ņemsim patvaļīgu vārdu (piemēram tukšo vārdu) un pārbaudīsim, vai dotā loģiskās shēmas reprezentācija to akceptē. Ja jā, tad skaidrs, ka tā atpazītā valoda nav tukša. Ja nē, tad skaidrs, ka ir vārdi, ko tas neakceptē. Tādā gadījumā pārbaudīsim, vai tam eksistē ekvivalenta GDA loģiskās shēmas reprezentācija, kuras BC-sarežģītība ir 0. Tā eksistē tad un tikai tad, ja dotā GDA pazītā valoda ir tukšā valoda.

Pēdējais (ceturtais) aplūkotais jautājums 9.5. teorēmā patiesībā ir GDA optimālas loģiskās shēmas atrašanas uzdevums, kas noformulēts kā atrisināmības problēma (decision problem), lai to varētu pieskaitīt kādai algoritmu sarežģītības klasei. Tas parāda, ka, lai arī standarta reprezentācijā minimālā GDA atrašana ir viegli (polinomiālā laikā) atrisināms uzdevums, optimālas loģiskās shēmas reprezentācijas atrašana ir PSPACE-pilnīga.

Bet šis jautājuma formulējums nav pats dabiskākais. Ja mēs atgriežamies pie stāvokļu kodēšanas uzdevuma, tad parasti tas tiek saprasts šādi: dotam GDA standarta (stāvokļu pāreju tabulas) reprezentācijā atrast tā optimālu (minimālu) loģiskās shēmas reprezentāciju.

Daudz kur literatūrā par šo uzdevumu ir teikts, ka tas ir NP-grūts[44][1] vai pat NP pilnīgs[3][2], kaut gan nevienā no tiem nav atrodams pat kaut kas līdzīgs šī apgalvojuma precīzam formulējumam, nemaz nerunājot par pierādījumu. Visdrīzāk autori ar to ir domājuši neformālu spriedumu, ka šī uzdevuma atrisināšanā nevar iztikt bez pilnās pārlases, tāpēc tas ir grūts. Bet mēs aplūkosim šo jautājumu formāli, pārveidosim to par atrisināmības uzdevumu (eksistenciālā formā) un pirmkārt jau ievērosim, ka, to var noformulēt divos dažādos veidos:

1. Standarta reprezentācijā (ar stāvokļu pāreju tabulu) uzdotam GDA noskaidrot, vai tā BC-sarežģītība ir mazāka par dotu skaitli k.
2. Standarta reprezentācijā (ar stāvokļu pāreju tabulu) uzdotam GDA noskaidrot, vai tā pazītās valodas BC-sarežģītība ir mazāka par dotu skaitli k.

Pirmajā variantā mums nepieciešams atrast minimālo loģiskās shēmas reprezentāciju tieši dotajam GDA, bet otrajā derēs arī jebkurš ekvivalents GDA. Šie abi jautājumi noteikti pieder klasei PSPACE. Dotam GDA mēs vispirms varam nedeterminēti atrast kādu loģiskās shēmas reprezentāciju ar sarežģītību ne lielāku par k, un tad pārbaudīt, ka tā tiešām reprezentē doto GDA, kā 9.5. teorēmas 4. gadījuma pierādījumā. Bet vismaz pirmajam uzdevumam to var izdarīt efektīvāk:

9.6. Teorēma *Uzdevums standarta reprezentācijā (ar stāvokļu pāreju tabulu) uzdotam GDA bez nesasniedzamiem stāvokļiem noskaidrot, vai tā BC-sarežģītība nepārsniedz dotu skaitli k, pieder sarežģītības klasei NP.*

Pierādījums Pieņemsim, ka mums ir dots GDA ar s stāvokļiem. Vispirms varam nedeterminēti uzminēt stāvokļu kodējumu un loģiskās shēmas reprezentāciju (ar BC-sarežģītību, kas nepārsniedz k), un tad pārliecināties, ka tā tiešām reprezentē doto GDA. Lai par to pārliecinātos, mums pietiek pārbaudīt visu dotā GDA stāvokļu pārejas tabulu, ko var izdarīt polinomiālā laikā attiecībā pret šīs pašas stāvokļu pārejas tabulas izmēru (kas ir arī ieejas garums). ∎

Pirmajā brīdī liekas, ka šis pats pierādījums derētu arī 2. uzdevumam, kur mums ir jāatrod BC-sarežģītība dotai valodai. Mēs varam nedeterminēti uzminēt dotajam GDA ekvivalentu GDA, tā kodējumu un loģiskās shēmas reprezentāciju, un tad polinomiālā laikā pārbaudīt, ka šie GDA ir ekvivalenti, un ka uzminētā loģiskās shēmas reprezentācija tiešām reprezentē šo GDA. Bet problēma rodas no tā, ka nav zināms, cik liels var būt šis ekvivalentais GDA. Labi būtu, ja tas būtu ierobežots ar kādu polinomu no dotā GDA izmēra, bet to mēs garantēt nevaram.

Piemēram, 7.1. teorēmas pierādījumā konstruētajam GDA A'_n eksistē ekvivalents GDA A_n, kura BC-sarežģītība ir krietni mazāka, bet tā stāvokļu skaits ir polinomiāli lielāks par A'_n stāvokļu skaitu. Pie tam šis polinoms ir atkarīgs no aplūkotās Tjūringa mašīnas stāvokļu skaita, tātad tas nav fiksēts.

Tādējādi jautājums, cik grūts tad īsti ir stāvokļu kodēšanas uzdevums, izrādās stipri netriviāls. Pirmajā formulējumā (kad jāatrod optimāla loģiskās shēmas reprezentācija dotam GDA), zināms, ka tas pieder klasei NP, bet nav nekā, kas norādītu, ka tas ir NP-pilnīgs, drīzāk otrādi. Varam uzskatīt, ka šis uzdevums sastāv no divām daļām: atrast dotajam GDA optimālu stāvokļu kodējumu un tad pie šī kodējuma atrast optimālu loģiskās shēmas reprezentāciju. Ja mēs aplūkojam tikai otro daļu, tad zināms[21], ka tas pieder klasi NP, bet zināms arī, ka gan tas, ja tas izrādītos piederīgs klasei P, gan arī tas, ja tas izrādītos NP-pilnīgs, novestu pie diezgan pārsteidzošiem secinājumiem.

Otrajā formulējumā šis uzdevums šķiet vēl grūtāks, jo nav pat zināms, vai tas pieder klasei NP.

Nobeigumā atzīmēsim, ka, ja mums ir dots nevis GDA, bet GNA, tad šis jautājums ir PSPACE-pilnīgs. Patiešām, ir labi zināms, ka noskaidrot, vai dotais GNA ir ekvivalents ar visu-akceptējošu GNA, ir PSPACE-pilnīgs uzdevums[15]. Tātad šis uzdevums GNA gadījumā ir PSPACE-pilnīgs pat $k = 0$ gadījumā.

10. nodaļa

Līdzīgi jēdzieni

Runājot ar cilvēkiem un stāstot par GDA loģiskās shēmas sarežģītību, pa reizei nākas dzirdēt jautājumus: “Bet vai tas nav tas pats kas ... ?” vai “ar ko tas atšķiras no ...?” un šie jautājumi mēdz atkārtoties. Tāpēc šajā nodaļā aplūkosim trīs jēdzienus, kas pirmajā brīdī liekas līdzīgi GDA loģiskās shēmas sarežģītībai un var izraisīt šādus jautājumus. Tā kā šai nodaļai ir vairāk paskaidrojošs raksturs, tad teorija šeit ir aprakstīta virspusēji, bet galvenais uzsvars ir likts uz atšķirību parādīšanu starp attiecīgo jēdzienu un GDA loģiskās shēmas reprezentāciju vai BC-sarežģītību.

10.1. Galīgi alternējoši automāti

Alternācija kā nedeterminisma vispārinājums parādījās teorētiskajā datorzinātnē 70-gadu beigās un jau pašos tās izpētes pirmsākumos tika aplūkoti arī galīgi alternējoši automāti[12][11]. Tam paralēli Leiss[25][26] savos rakstos aplūkoja Būla automātus, kas pēc savas būtības ir tie paši alternējošie galīgie automāti, tikai tiem ir pieļaujami vairāki sākuma stāvokļi.

Galīgi alternējoši automāti (GAA) ir loģisks GNA vispārinājums. Ja no kāda GNA iziet vairākas pārejas, tad to rezultātiem tiek pielietota loģiskā disjunkcija. Piemēram, ja GNA no stāvokļa q_1, nolasot ieejas simbolu a, var nokļūt stāvokļos q_2 un q_3, tad šis GNA stāvoklī q_1 akceptē vārdu aw, ja tas akceptē vārdu w stāvoklī q_2 VAI stāvoklī q_3. GAA šīs disjunkcijas vietā var būt patvaļīga Būla funkcija, t.i. GAA ar n stāvokļiem ($|Q| = n$) alfabētā Σ pārejas funkcija ir

$$\delta : Q \times \Sigma \to (\{0, 1\}^n \to \{0, 1\}) .$$

Tādējādi katram GAA stāvoklim un katram ieejas simbolam atbilst Būla funkcija $\{0, 1\}^n \to \{0, 1\}$. Tas pirmajā brīdī liekas līdzīgi GDA loģiskās shēmas reprezentācijas stāvokļu bitiem: katram (izejas) bitam, katram ieejas simbolam arī atbilst Būla funkcija no visiem stāvokļu bitiem. Bet, aplūkojot abas situācijas sīkāk, izrādās, ka līdzība šeit ir tikai vizuāla. GDA loģiskās shēmas gadījumā šīs Būla funkcijas atbilst izejas bitiem (un ieejas simboliem): katru izejas bitu var izrēķināt ar Būla funkciju. Bet GAA gadījumā šīs funkcijas atbilst ”ieejas bitiem” t.i. stāvoklim, kurā automāts dotajā brīdī atrodas, un tās apraksta stāvokli, uz kuru tas pāriet.

Arī pats stāvokļa jēdziens GDA loģiskās shēmas reprezentācijai un GAA ir būtiski atšķirīgs: Ja GDA loģiskās shēmas reprezentācijā GDA stāvoklim atbilst tā kodējums, t.i. stāvokļa bitu konkrēta vērtība, tad GAA gadījumā automāta stāvoklim dotajā brīdī atbilst patvaļīga Būla funkcija no tā stāvokļiem.

Līdzīgi kā GNA, arī GAA var izrēķināt tikai regulāras valodas, bet dažiem GAA ar n stāvokļiem atbilstošajam minimālajam GDA ir 2^{2^n} stāvokļu. Turpretī, jebkurai GDA loģiskās shēmas reprezentācijai ar n stāvokļu bitiem atbilstošajam GDA ir ne vairāk par 2^n stāvokļiem.

Bet kaut kāda līdzība starp šiem jēdzieniem tomēr pastāv un būtu interesanti izpētīt GAA no loģiskās shēmas sarežģītības viedokļa. Šīs līdzības ilustrācijai aplūkosim teorēmu no[25]:

10.1. Teorēma *Valodu L var akceptēt ar GDA ar 2^n stāvokļiem tad un tikai tad, ja L^R var akceptēt ar GAA ar $n+1$ stāvokli.*

Redzams, ka, ja L^R vietā būtu L, tad šeit būtu līdzība ar GDA loģiskās shēmas reprezentāciju, kur par tās sarežģītību būtu ņemts stāvokļa bitu skaits: katram kodējumam ar n stāvokļu bitiem atbilst GDA ar 2^n stāvokļiem un otrādi.

10.2. Regulāru valodu loģiskās shēmas sarežģītība

Šī darba galvenais jēdziens — BC-sarežģītība tā saucas tāpēc, ka daudz loģiskāks nosaukums "loģiskās shēmas sarežģītība" jau ir aizņemts. Saka, ka loģisko shēmu saime $\{c_i\}$ pazīst valodu L, ja visiem vārdiem x garumā n

$$x \in L \leftrightarrow c_n(x) = 1,$$

un valodas L loģiskās shēmas sarežģītība nepārsniedz $f(n)$, ja to pazīst loģisko shēmu saime $\{c_i\}$, un $C(c_n) \leq f(n)$ visiem $n \in \mathbb{N}$.

Valodu loģiskās shēmas sarežģītība jau tika pieminēta 2.6. nodaļā, runājot par sarežģītības klasi P/poly un Karpa-Liptona teorēmu. Sarežģītības klase P/poly satur visas valodas, kuru loģiskās shēmas sarežģītība ir ierobežota ar kādu polinomu. Šī klase ir liela, acīmredzami, ka P∈P/poly, jo jebkuru Tjūringa mašīnu, kas darbojas polinomiālā laikā, var aprakstīt ar polinomiāla izmēra loģisko elementu shēmu (2.8. teorēma). Bet tai pieder pat daudzas nerekursīvas valodas. Jebkurai binārai valodai L mēs varam nodefinēt valodu

$$Long(L) = \{x : |x| = n \text{ un } n \in L\},$$

kas katram garumam n saturēs vai nu visus vārdus vai arī nevienu, atkarībā no tā, vai n pieder valodai L vai ne. Acīmredzami, ka $Long(L)$ nav rekursīva, ja L nav rekursīva, bet $Long(L)$ loģiskās shēmas sarežģītība ir konstanta, jo to pazīst loģisko shēmu saime $\{c_i\}$, kur katra c_i atgriež vai nu konstantu 0 vai konstantu 1.

Visas regulārās valodas acīmredzami arī pieder klasei P/poly (jo tās atpazīstošo GDA var uztvert kā TM, kas darbojas lineārā laikā), bet to var pētīt arī sīkāk. Regulāru valodu loģiskās shēmas sarežģītība tiek aplūkota daudzos rakstos un grāmatās[36], kur, izmantojot regulāras valodas sintaktisko monoīdu, tiek pamatots, ka visas regulāras valodas pieder klasei NC^1 (valodas, ko var vienmērīgi pazīt ar logaritmiska dziļuma loģiskajām shēmām), pie tam tās, kuru sintaktiskais monoīds nav atrisināms, ir šajā klasē pilnīgas.

10.3. Saistība ar Kolmogorova sarežģītību

Kā alternatīvu BC-sarežģītībai šajā nodaļā aplūkosim galīgu automātu Kolmogorova sarežģītību. Tām abām ir kaut kas kopīgs — tās mēģina aprakstīt automātu pēc iespējas kompaktāk. Galvenā atšķirība ir tā, ka Kolmogorova sarežģītība rūpējas tikai par automāta aprakstu, bet BC-sarežģītība — arī par to, lai šo automātu no šāda apraksta varētu izpildīt, t.i. lai katrai ieejai un stāvoklim efektīvi varētu atrast nākamo stāvokli. Kolmogorova sarežģītība turpretī mēģina saspiest automātu pēc iespējas mazāku un nerūpējas par to, cik ilgā laikā to var iegūt atpakaļ un cik ilgā laika ar to varēs apstrādāt (pazīt vai noraidīt) kādu vārdu.

Definīcija Par galīga automāta A Kolmogorova sarežģītību sauksim minimālo skaitli $n = K(A)$, tādu, ka eksistē Tjūringa mašīna M, kuras izmērs ir n un kura, saņemot ieejā tukšu lenti, izejā izdod automāta A bināro pierakstu.

Acīmredzami, ka Kolmogorova sarežģītība, salīdzinot ar BC-sarežģītību (un arī stāvokļu sarežģītību), var būt neierobežoti mazāka.

10.2. Teorēma *Patvaļīgai visur definētai augošai daļēji rekursīvai funkcijai, var atrast tādu galīgu automātu virkni A_n^f, ka $C_{\text{BC}}(A_n^f) > f(K(A_n^f))$ pietiekami lieliem n.*

Pierādījums Uzrakstīsim programmu $\mathfrak{M}(f,n)$, kas saņem ieejā patvaļīgu daļēji rekursīvu funkciju $f : \mathbb{N} \to \mathbb{N}$ un skaitli n un kas izdrukā stāvokļu pārejas tabulu GDA A_n^f viena simbola alfabētā, kuram ir $2^{f(2n)}$ stāvokļu, kura pārejas funkcija ir "riņķis", kas iet caur visiem stāvokļiem un kurš akceptē vārdus tikai sākumstāvoklī (tātad tas akceptē visus vārdus garumā $2^{f(2n)}$).

Acīmredzami, ka izdrukātais automāts A_n^f ir minimāls, tā stāvokļu sarežģītība ir $s = 2^{f(2n)}$ un tā BC-sarežģītība pēc 4.7. teorēmas apakšējās robežas nav mazāka par $\lceil \log s \rceil = f(2n)$.

Bet tā Kolmogorova sarežģītību, izmantojot S-m-n teorēmu, var novērtēt kā

$$K(A_n^f) \leq K(f) + K(n) + c \leq n + c',$$

kur c, c' ir konstantes (kas var būt atkarīgas no f). Līdz ar to pietiekami lieliem n (kad $n > c'$):

$$C_{\text{BC}}(A_n^f) = f(2n) > f(n + c') \geq f(K(A_n^f)). \quad \blacksquare$$

Pretstatā BC-sarežģītībai, kurai var gadīties, ka minimālajam automātam tā ir vairāk nekā polinomiāli lielāka nekā kādam citam ekvivalentam automātam, Kolmogorova sarežģītība šajā ziņā ir lineāri ierobežota.

10.3. Teorēma *Eksistē konstante c, tāda, ka jebkuram automātam A*

$$K(M(A)) \leq K(A) + c$$

Pierādījums Ņemsim Tjūringa mašīnu N ar $K(A)$ stāvokļiem, kas izejā izdod automāta A bināro pierakstu. Vēl ņemsim Tjūringa mašīnu M, kas izpilda minimizācijas algoritmu galīgiem automātiem (ieejā tā saņem patvaļīgu galīga automāta bināro pierakstu un izejā izdod atbilstošā minimālā automāta bināro pierakstu).

Tad Tjūringa mašīna $M(N)$, ko iegūst, izpildot M uz N izdoto izeju, pārveido tukšu lenti par minimālo automātu M(A). Ja TM X stāvokļu skaitu apzīmē ar $S(X)$, tad tās stāvokļu skaits ir S(M)+S(N), tātad

$$K(M(A)) \leq S(N(M)) = S(N) + S(M) = K(A) + c$$

kur $c = S(M)$. $\blacksquare$

11. nodaļa

Secinājumi

Šajā disertācijā no dažādiem skatu punktiem aplūkota GDA BC-sarežģītība, ko var uztvert kā formālismu GDA stāvokļu kodēšanas uzdevumam. Lai arī tās definīcija ir vienkārša un dabiska un to būtu varēts pētīt jau 60. gados, kad radās un attīstījās automātu teorija, tomēr līdz šim no šāda (sarežģītības) skatupunkta tas netika aplūkots.

BC-sarežģītība pēc būtības ir GDA pārejas funkcijas izteiktas kā loģisko elementu shēmas sarežģītība. Šajā darbā tā ir analizēta gan no aprakstošās gan algoritmiskās sarežģītības viedokļa.

BC-sarežģītība savā veidā novērtē GDA strukturālo sarežģītību: gadījumos, ja GDA nav iekšējas struktūras, tā ir tuva stāvokļu sarežģītībai, bet, ja ir, tad tā var būt pat eksponenciāli mazāka (4.7. teorēma).

4. nodaļā tika aplūkotas regulāru valodu BC-sarežģītības augšējās un apakšējās robežas attiecībā pret to stāvokļu sarežģītību. Tika pierādīts tā sauktais "Šenona efekts" BC-sarežģītībai: gandrīz visām valodām BC-sarežģītība ir tuva savai maksimālajai vērtībai $(k-1)s$ (4.11. teorēma), kur k ir simbolu skaits alfabētā, bet s — automāta stāvokļu skaits (ja $k=1$, tad maksimālā vērtība ir $\frac{s}{\log s}$).

Tālāk 5. nodaļā šis pats jautājums tika aplūkots nedeterminētās stāvokļu sarežģītības gadījumā. Šajā gadījumā augšējās un apakšējās robežas BC-sarežģītībai attiecībā pret nedeterminēto stāvokļu sarežģītību lai arī ir diezgan tuvas ($k \geq 2$ gadījumā atšķiras ne vairāk kā 4 reizes), tomēr nesakrīt. Tāpēc pagaidām jautājums, vai Šenona efekts ir spēkā BC-sarežģītībai valodām ar dotu nedeterminēto stāvokļa sarežģītību, paliek atklāts.

Tāpat var ievērot, ka gan stāvokļa sarežģītības, gan nedeterminētās stāvokļa sarežģītības gadījumā BC-sarežģītības augšējās un apakšējās robežas ir diezgan līdzīgas: apakšējās robežas sakrīt, bet augšējās atšķiras kvadrātiski. Līdz ar to kaut kādā ziņā automāta realizācijai ar loģisko shēmu nav pārāk būtiski, vai sākotnējais automāts ir determinēts vai nedeterminēts, tā loģiskās shēmas sarežģītība atradīsies līdzīgās robežās.

Aplūkoti arī BC-sarežģītības novērtējumi klasiskākajām valodu operācijām. Valodu operācijām rezultāti ir doti tikai vairāku simbolu alfabēta gadījumā, viena simbola alfabēta gadījumā tie var atšķirties. Šīm teorēmām (6.2. — 6.5.) trūkst arī atbilstošo apakšējo novērtējumu, lai gan tas nešķiet vienkāršs uzdevums, jo tas savā ziņā ietver apakšējos novērtējumus loģisko shēmu sarežģītībai.

Viens no, manuprāt, pašiem interesantākajiem šī darba rezultātiem atrodams

7. nodaļā. Tajā stingrā formā pamatots vispārzināmais fakts, ka GDA minimizācija var novest pie tās struktūras "sarežģīšanās". Līdz šim šis fakts tika pamatots ar nelieliem piemēriem, bet 7.1. teorēma apgalvo, ka regulāras valodas minimālā GDA BC-sarežģītība nav polinomiāli ierobežota ar pašas valodas BC-sarežģītību.

Taču cits jautājums, kas ir kaut kādā ziņā līdzīgs iepriekšējam, ir palicis neatrisināts: vai BC-sarežģītība var samazināties, ja izmanto kodējumu, kas nav minimālais? Zināms, ka n stāvokļu GDA stāvokli var iekodēt $\lceil \log n \rceil$ stāvokļa bitos. Vai var gadīties, ka šī GDA minimālajai loģiskās shēmas reprezentācijai (reprezentācijai ar minimālo BC-sarežģītību) ir vairāk par $\lceil \log n \rceil$ stāvokļa bitiem?

Daudzi uzdevumi, kas GDA standarta reprezentācijas gadījumā ir tik viegli, ka tos reti kad aplūko (stāvokļu sasniedzamība, ekvivalence), izrādās daudz grūtāki gadījumā, ja GDA ir uzdots ar loģisko elementu shēmu. 9. nodaļā pamatots, ka vairāki šādi jautājumi, tai skaitā BC-sarežģītības minimizācija, ir PSPACE-pilnīgi.

Taču, iespējams, vispraktiskākais jautājums no algoritmiskās sarežģītības viedokļa: cik sarežģīti ir dotam GDA (standarta reprezentācijā) atrast optimālu loģiskās shēmas reprezentāciju, pagaidām paliek neatrisināts.

Literatūra

[1] I Ahmad and MK Dhodhi. State assignment of finite-state machines. *IEE Proceedings-Computers and Digital Techniques*, 147(1):15–22, 2000.

[2] Walid M Aly. Solving the state assignment problem using stochastic search aided with simulated annealing. *Am. J. Eng. Appl. Sci*, 2(4):710–714, 2009.

[3] José Nelson Amaral, Kagan Tumer, and Joydeep Ghosh. Designing genetic algorithms for the state assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):687–694, 1995.

[4] José L Balcázar, Antoni Lozano, and Jacobo Torán. The complexity of algorithmic problems on succinct instances. In *International Conference of the Chilean Computer Science Society*, pages 351–377. Springer, 1992.

[5] Luca Benini and Giovanni De Micheli. State assignment for low power dissipation. *IEEE Journal of Solid-State Circuits*, 30(3):258–268, 1995.

[6] Ravi B Boppana and Michael Sipser. *The complexity of finite functions*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1989.

[7] Bernd Borchert and Antoni Lozano. Succinct circuit representations and leaf languages are basically the same concept. Technical report, Universitat Politechnica de Catalunya, 1996. `http://upcommons.upc.edu/handle/2117/97245`.

[8] Ney Laert Vilar Calazans. Considering state minimization during state assignment. In *I Ibero American Microelectronics Conference-X Congress of the Brazilian Microelectronics Society*, pages 49–58.

[9] Ney LV Calazans. *State minimization and state assignment of finite state machines: their relationship and their impact on the implementation*. PhD thesis, PhD thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1993.

[10] Cezar Campeanu, Karel Culik, Kai Salomaa, and Sheng Yu. State complexity of basic operations on finite languages. In *International Workshop on Implementing Automata*, pages 60–70. Springer, 1999.

[11] Ashok K Chandra, Dexter C Kozen, and Larry J Stockmeyer. Alternation. *Journal of the ACM (JACM)*, 28(1):114–133, 1981.

[12] Ashok K Chandra and Larry J Stockmeyer. Alternation. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 98–108. IEEE, 1976.

[13] Giovanni De Micheli, Robert K Brayton, and Alberto Sangiovanni-Vincentelli. Optimal state assignment for finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 4(3):269–285, 1985.

[14] Michael Domaratzki, Derek Kisman, and Jeffrey Shallit. On the number of distinct languages accepted by finite automata with n states. *Journal of Automata, Languages and Combinatorics*, 7(4):469–486, 2002.

[15] Michael R Garey and David S Johnson. *Computers and intractability*. Freeman, 1979.

[16] Yo-Sub Han and Kai Salomaa. State complexity of basic operations on suffix-free regular languages. *Theoretical Computer Science*, 410(27-29):2537–2548, 2009.

[17] Juris Hartmanis and Richard Edwin Stearns. Some dangers in state reduction of sequential machines. *Information and Control*, 5(3):252–260, 1962.

[18] Markus Holzer and Martin Kutrib. Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science*, 14(06):1087–1102, 2003.

[19] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *ACM SIGACT News*, 32(1):60–65, 2001.

[20] Neil D Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975.

[21] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 73–79. ACM, 2000.

[22] Krzysztof Kajstura and Dariusz Kania. Low power synthesis of finite state machines—state assignment decomposition algorithm. *Journal of Circuits, Systems and Computers*, page 1850041, 2017.

[23] Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309. ACM, 1980.

[24] Zvi Kohavi and Niraj K Jha. *Switching and finite automata theory*. Cambridge University Press, 2009.

[25] Ernst Leiss. Succinct representation of regular languages by boolean automata. *Theoretical computer science*, 13(3):323–330, 1981.

[26] Ernst Leiss. Succinct representation of regular languages by boolean automata ii. *Theoretical Computer Science*, 38:133–136, 1985.

[27] Antonio Lozano and José L Balcázar. The complexity of graph problems for succinctly represented graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 277–286. Springer, 1989.

[28] OB Lupanov. *Asymptotic Estimates of the Complexity of Control Systems (in Russian)*. Moscow State University, 1984.

[29] George H Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[30] Frank R Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transactions on computers*, 100(10):1211–1214, 1971.

[31] RG Nigmatullin. *The complexity of Boolean functions (in russian)*. Nauka, Moscow, 1991.

[32] Michael O Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

[33] Arto Salomaa. *Theory of automata*. Elsevier, 2014.

[34] Arto Salomaa, Kai Salomaa, and Sheng Yu. State complexity of combined operations. *Theoretical Computer Science*, 383(2-3):140–152, 2007.

[35] Claude Shannon et al. The synthesis of two-terminal switching circuits. *Bell Labs Technical Journal*, 28(1):59–98, 1949.

[36] Howard Straubing. *Finite automata, formal logic, and circuit complexity*. Springer Science & Business Media, 2012.

[37] Robert Endre Tarjan. Complexity of monotone networks for computing conjunctions. *Annals of Discrete Mathematics*, 2:121–133, 1978.

[38] BA Trachtenbrot and JM Barzdin. *Finite Automata: Synthesis and Behaviour*. Nauka, Moscow, 1970.

[39] Māris Valdats. Boolean circuit complexity of finite automata. In *Computer science and information technologies*, pages 63–67. National academy of sciences of Armenia, 2011.

[40] Māris Valdats. Transition function complexity of finite automata. In *International Workshop on Descriptional Complexity of Formal Systems*, pages 301–313. Springer, 2011.

[41] Māris Valdats. Boolean circuit complexity of regular languagues. In *International Conference on Automata and Formal Languages*, volume 151, pages 342–354. EPTCS, 2014.

[42] Māris Valdats. Descriptional and computational complexity of the circuit representation of finite automata. In *International Conference on Language and Automata Theory and Applications*, pages 105–117. Springer, 2018.

[43] Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.

[44] Wayne Wolf, Kurt Keutzer, and Janaki Akella. A kernel-finding state assignment algorithm for multi-level logic. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pages 433–438. IEEE Computer Society Press, 1988.

[45] Sheng Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221, 2001.

[46] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.