Tallinna Tehnikaülikool
Eesti TA Küberneetika Instituut

Teoreetiline informaatika
I osa

Loengukonspekt CS 70/94

Jaan Penjam

Tallinn • Jaanuar 1995

Sisukord

1. Sissejuhatus

Teoreetiline informaatika on distsipliin, mis on kujunenud mitme erineva arvutitega seotud teadus- ja tegevusvaldkonna - programmeerimise, diskreetse matemaatika, kompuuterlingvistika, elektroonika jms. - piirimail. Kursuse aineks on arvutiteaduse ja tarkvaratehnika mudelid ja meetodid, algoritmide ja arvutusprotsesside kõige üldisemad seaduspärasused, programmide ning andmete struktuur ning semantika.

Kursuse eesmärgiks on kuulajaile tutvustada:

- abstraktsiooni olemust ja rolli arvutiteaduses;
- peamisi arvutuste mudeleid;
- formaalset matemaatilist aparatuuri ja meetodeid analüüsimaks arvutusprotsesside kulgu, programmide korrektsust ja efektiivsust;
- programmide konstrueerimise meetodeid.

Informaatika põhimeetod tegelikkuse tunnetamisel on modelleerimine, s.o. objektide mudelite koostamine ja nende kasutamine. Mudel on objekt, mis on kindlas (analoogia seose alusel) vastavuses uuritava objektiga (originaaliga). Eristatakse geomeetrilisi, analoog- ja matemaatilisi mudeleid. Informaatikas on eelkõige tegu viimatinimetatutega. Kuna matemaatiline mudel on alati abstraktne, saadud abstraktsiooni tulemusena, sellest siis ka abstraktsiooni tähtsus informaatika aluste uurimisel.

Informaatika teoreetiliste aluste klassikaline kursus sisaldab endas järgmisi peatükke:

- hulgateooria ja konstruktiivne loogika;
- formaalsete keelte teooria;
- rekursiooniteooria ja algoritmianalüüs;
- klassikaliste andmetöötlusülesannete lahendusalgoritmid.

Kuna mitmeid nimetatud küsimustest käsitletakse meie kõrgkoolis põhjalikult diskreetse matemaatika (hulga- ja graafiteooria) ja mitmete erikursuste raames (otsimis- ja sorteerimisalgoritmid, formaalloogika, programmide konstrueerimine), on käesolev kursus üles ehitatud mõnevõrra erinevana. Peamine rõhuasetus on tehtud formaalsete keelte teooriale, lahenduvuse ja algoritmide keerukuse küsimustele. Tehakse ka lühike sissejuhatus programmide semantikasse, konstrueerimisse ja verifitseerimisse. Viimatinimetatud valdkonna detailsem käsitlus jääb aga siiski stuudiumi lõpupoole - neljandaks -viiendaks õppeaastaks ning magistriõppe raames toimuvateks erikursusteks. Teoreetilise informaatika ainetsükli terviklikuks omandamiseks on soovitav kuulata veel TTÜ-s esitatavaid kursusi "loogika arvutiteaduses", "automaatne programmeerimine" ja "teoreetiline informaatika II".

Järgnevalt toome käesoleva kursuse raames planeeritud loenguteemade loetelu.

1. Algoritmilised keeled.
2. Regulaarsed struktuurid.
3. Kontekstivabad struktuurid.
4. Rekursiooniteooria.
5. Algoritmide keerukus.
6. Paralleelprogrammid.

2. Eelteadmised matemaatikast (kordamine)

2.1. Hulgateooria

Teoreetilise informaatika mudelite esitamiseks kasutatakse peamiselt matemaatilisi formalisme. Kõige enam on vaja hulgateooria mõisteid ning tähistusviisi ja graafiteooriat. Järgnevas vaatlemegi lühidalt nimetatud valdkondade põhimõisteid.

Hulk on samalaadsete objektide järjestamata kogum, mida teatud konkreetses kontekstis käsitletakse kui tervikut. Hulka kuuluvaid objekte nimetatakse hulga elementideks. Elemendi a kuulumist hulka A tähistatakse $a \in A$. Hulkade esitamiseks kasutatakse kaht moodust:

1. elementide loetlemist, näiteks
$$A = \{1, 4, 7, 9, 12\}$$
või
$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, ...\};$$
2. predikaadi abil, s.t avaldisega kujul $H = \{z|P(z)\}$, näiteks
$$L = \{x|3x^2 - 4x + 2 = 0\}.$$

Tühja hulga tähis on $\emptyset$.

Lõpliku hulga elementide arvu nimetatakse hulga võimsuseks. Hulga A võimsust tähistatakse sümboliga $|A|$. Lõpmatute hulkade võimsus määratakse kindlaks tema elementide seadmisel üks-ühesesse (*bijektiivsesse*) vastavusse tuntud hulkadega, näiteks naturaalarvude hulgaga $\mathbb{N}$.

Predikaadi abil hulga määramisel võib tekkida vastuolu, nii nagu näiteks järgmise Russelli paradoksi puhul.

Tähistagu predikaat $P(X)$ tingimust, mis on rahuldatud (on tõene), kui argumendina antud hulk X pole iseenda element ja väär vastasel juhul. Hulga $Y = \{X|P(X)\}$ korral predikaadi P kontrollimine viib vastuoluni:

- juhul, kui oletame, et $Y \in Y$, siis $P(Y)$ on väär ning hulga Y määratluse põhjal $Y \notin Y$;
- kui oletame, et $Y \in Y$, siis $P(Y)$ on tõene ning seega $Y \in Y$.

Kirjeldatud vastuolu on üks paljudest hulgateooria paradoksidest, mis viitavad selle teooria ebatäiuslikkusele. Toodud paradoks on filosoofilises kirjanduses laiemalt tuntud Russelli habemeajaja paradoksi nime all.

Hulkade võrdlemine:

- $A = B$ - hulgad A ja B on võrdsed (sisaldavad samu elemente);
- $A \subseteq B$ - hulk A on hulga B osahulk (võib ka võrduda A-ga);
- $A \subset B$ - hulk A on hulga B pärisosahulk ($A \neq B$).

Hulga A kõigi alamhulkade hulka tähistatakse kas $\mathcal{P}(A)$ või 2^A. Kui $|A| = n$, siis $|2^A| = 2^n$.

Tehted hulkadega:

- $A \cup B$ - hulkade A ja B ühend;
- $A \cap B$ - hulkade A ja B ühisosa e. lõige;
- $A \setminus B$ või $A - B$ - hulkade A ja B vahe;
- A' - hulga A täiend;
- $A \times B$ -rist- e. otse- e. Descartes'i korrutis:
$$A \times B = \{(a, b)|a \in A \& b \in B\}$$

2.2. Relatsioonid e. seosed e. suhted

- Seos hulkade A ja B vahel on alamhulk $R \subseteq A \times B$.
- Seos hulgal A on alamhulk $R \subseteq A \times A$.
- Asjaolu, et $(a,b) \in R$ tähistatakse aRb. Seost $R^{-1} = \{(b,a)|aRb\}$ nimetatakse seose R *pöördrelatsiooniks.*

Definitsioon 1. *Seost R hulgal A nimetatakse* ekvivalentsiseoseks *siis ja ainult siis, kui*

- R *on* refleksiivne, *s.t.* $\forall a \in A$ *korral* aRa;
- R *on* sümmeetriline, *s.t.* $a,b \in A$ *korral* $aRb \Longrightarrow bRa$;
- R *on* transitiivne, *s.t.* $a,b,c \in A$ *korral* $aRb \& bRc \Longrightarrow aRc$.

Definitsioon 2. *Olgu R ekvivalentsiseos hulgal A. Elemendiga $a \in A$ ekvivalentsete elementide hulka*

$$[a] = \{b|aRb\}$$

nimetatakse elemendi a ekvivalentsiklassiks.

Näide 1. Jäägiklassid naturaalarvude hulgal $\mathbb{N}$.

Vaatleme seost "arv $a \in \mathbb{N}$ on võrdne arvuga $b \in \mathbb{N}$ mooduli N suhtes", mis kehtib parajasti siis kui $\exists k \in \mathbb{N}$, nii et $|a-b| = kN$. Seda seost tähistatakse $a \equiv b(modN)$. Lihtne on näha, et vaadeldav seos on ekvivalentsiseos naturaalarvude hulgal $\mathbb{N}$. Juhul kui $N = 3$ on arvude 0, 1 ja 2 ekvivalentsiklassid (antud juhul nimetatakse neid arvu 3 jäägiklassideks) järgmised:

$$[0] = 0,3,6,9,12,...;$$

$$[1] = 1,4,7,10,13,...;$$

$$[2] = 2,5,8,11,14,....$$

Nagu näha, jäägiklassid omavahel ei lõiku, kuid katavad täielikult kogu naturaalarvude hulga. Nagu järgmisest teoreemist selgub, kehtib selline seos mistahes ekvivalentsiseose korral.

□

Teoreem 1. *Olgu R ekvivalentsiseos hulgal A. Iga $a \in A$ ja $b \in A$ korral kehtib kas $[a] = [b]$ või $[a] \cap [b] = \emptyset$.*

Tõestus. Vaatleme algul juhtu, kus kehtib aRb. Valime klassist $[a]$ vabalt ühe elemendi z. Siis kehtivad bRa (ekvivalentsiseose sümmeetrilisuse tõttu) ja aRz, millest ekvivalentsiseose transitiivsuse omaduse põhjal järeldub, et bRz. Seega $z \in [b]$ ja järelikult $[a] \subseteq [b]$. Analoogiliselt saab näidata, et $[b] \subseteq [a]$. Kokkuvõttes, kui aRb, siis $[a] = [b]$. Juhul kui $\neg(aRb)$, eeldame vastuväiteliselt, et leidub selline $y \in A$, et $y \in [a]$ ja $y \in [b]$. Teisisõnu, kehtivad aRy ja bRy. Ekvivalentsiseose sümmeetrilisuse tõttu kehtib ka yRb, millest omakorda järeldub transitiivsuse põhjal, et aRb. Viimane tulemus on vastuolus tehtud eeldusega, et $(a,b) \notin R$. Vastuolu põhjuseks on väär eeldus, et käesoleval juhul pole klasside $[a]$ ja $[b]$ ühisosa tühi.

□

Definitsioon 3. *Olgu R seos hulgal A. Seose R* k-s aste *R^k on määratud järgmiste tingimustega:*

- aR^1b *kehtib parajasti siis, kui* aRb;
- aR^kb $(k > 1)$ *kehtib parajasti siis, kui* $\exists c \in A$, *nii et*

$$aRc \& cR^{k-1}b.$$

Definitsioon 4. *Olgu R seos hulgal A. Seose R* transitiivseks sulundiks *nimetatakse seost $R^+ \subseteq A \times A$, nii et aR^+b kehtib parajasti siis, kui $\exists i \geq 1$, mille korral kehtib aR^ib.*

Definitsioon 5. *Olgu R seos hulgal A. Seose R* refleksiivseks transitiivseks sulundiks *nimetatakse sellist seost R^* , nii et*

- *aR^*a iga $a \in A$;*
- *aR^*b kehtib siis, kui aR^+b;*
- *R^* ei sisalda rohkem elementide paare kui on määratud viimase kahe tingimusega.*

Teoreem 2. *Kui R^+ ja R^* on vastavalt seose R transitiivne ja refleksiivne-transitiivne sulund, siis*

- *Kui R' on suvaline transitiivne seos, selline et $R \subseteq R'$, siis $R^+ \subseteq R'$;*
- *Kui R' on suvaline refleksiivne-transitiivne seos, selline et $R \subseteq R'$, $R^* \subseteq R'$.*

2.2.1. Järjestusseosed

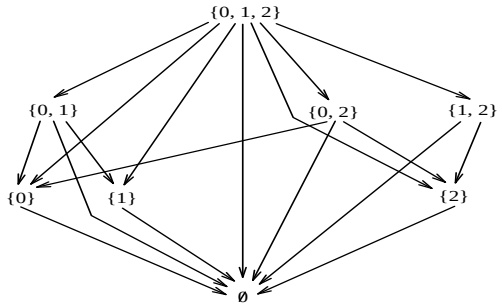
Definitsioon 6. *Seos R määrab hulgal A* osalise järjestuse, *kui*

- *R on transitiivne;*
- *R on irrefleksiivne, s.t. $\forall a \in A$ korral $(a, a) \notin R$.*

Näide 2. Vaatleme hulga A kõigi osahulkade hulgal $\mathcal{P}(A)$ alamhulga seost:

$$\alpha R \beta \text{ kehtib parajasti siis, kui } (\alpha \supset \beta) \& (\alpha \neq \beta).$$

Kui $A = \{0, 1, 2\}$, võib esitatud osalise järjestuse esitada ka graafiliselt.



□

Definitsioon 7. *Seos $R \subseteq A \times A$ on* refleksiivne osaline järjestus *hulgal A, kui*

- *R on transitiivne;*
- *R on refleksiivne;*
- *$(aRb) \& (bRa) \Longrightarrow (a = b)$.*

Definitsioon 8. *Refleksiivne seos $R \subseteq A \times A$ määrab hulgal A* lineaarse (lineaarse refleksiivse) *järjestuse, kui kehtib parajasti üks järgmistest tingimustest: aRb, bRa või $a = b$.*

Näiteks $<$ on lineaarne järjestus hulgal $\mathbb{N}$ ja $\leq$ on lineaarne refleksiivne seos hulgal N. Seevastu näites 2 esitatud järjestusseos pole lineaarne, kuna näiteks hulgad $\{1, 2\}$ ja $\{0, 2\}$ pole võrreldavad.

Rõhutamaks asjaolu, et hulgal A on määrarud järjestusseos R, kasutatakse hulga tähistamiseks avaldist (A, R). Näiteks $(\mathbb{N}, <)$, $(\mathbb{N}, \leq)$, $(\mathcal{P}(A), R)$ jne.

2.2.2. Kujutused

Definitsioon 9. *Kujutuseks M hulgast A hulka B nimetatakse seost $M \subseteq A \times B$, nii et $(a,b) \in M \& (a,c) \in M \Longrightarrow b = c$.*

Definitsioonis 9 esitatud kujutust tähistatakse $M : A \longrightarrow B$. Asjaolu, et $(a,b) \in M$ tähistatakse ka $M(a) = b$.

Kujutuse $M : A \longrightarrow B$ muutmispiirkond $\mathrm{Dom}(M)$ ja määramispiirkond $\mathrm{Ran}(M)$ on defineeritavad vastavalt seostega

$$\mathrm{Dom}(M) = \{a \mid a \in A,\ \exists b \in B(b = M(a))\}$$

ja

$$\mathrm{Ran}(M) = \{b \mid b \in B,\ \exists a \in A(M(a) = b)\}.$$

Definitsioon 10. *Kujutust $M : A \longrightarrow B$ nimetatakse*

- osaliseks, *kui* $\mathrm{Dom}(M) \subset A$, *s.t.* $\mathrm{Dom}(A)$ *on hulga A pärisosahulk;*
- täielikuks, *kui* $\mathrm{Dom}(M) = A$;
- pealekujutuseks *e.* sürjektsiooniks, *kui M on täielik ja* $\mathrm{Ran}(M) = B$;
- üks-üheseks *e.* injektsiooniks, *kui iga $a, a' \in A$ ning iga $b \in B$ korral kehtib seos*

$$(f(a) = b \& f(a') = b) \Longrightarrow a = a';$$

- bijektsiooniks, *kui ta on sürjektsioon ja injektsioon.*

Definitsioon 11. *Hulgad A ja B on* võrdvõimsad, *kui leidub bijektiivne kujutus $M : A \longrightarrow B$.*

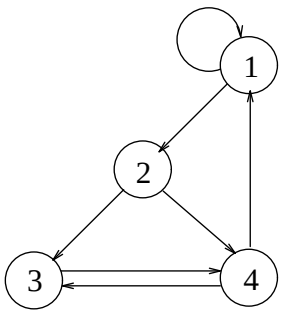
Definitsioon 12. *Lõpmatu hulk A on* loenduv, *kui ta on võrdvõimas naturaalarvude hulgaga* $\mathbb{N}$.

Loenduva hulga võimsust tähistatakse sümboliga ω, seega $|\mathbb{N}| = \omega$.

2.3. Graafiteooria

Definitsioon 13. *(Mittejärjestatud ja mitteorienteeritud)* graaf *on paar $G = (A, R)$, kus A on* tippude *hulk ja $R \subseteq A \times A$ seos hulgal A. Hulka R nimetatakse* kaarte hulgaks.

Näide 3. Graafi graafiline esitus: Olgu graaf $G = (A, R)$, kus $A = \{1, 2, 3, 4\}$; $R = \{(1,1),\ (1,2),\ (2,3),\ (2,4),\ (3,4)\ (4,1),\ (4,3)\}$.



□

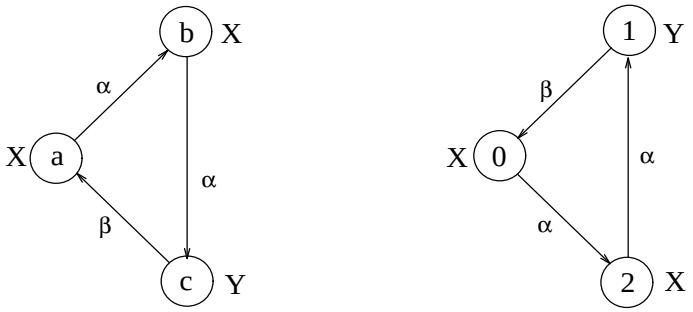
Definitsioon 14. *Graafid $G_1 = (A_1, R_1)$ ja $(G_2 = (A_2, R_2)$ on* võrdsed (isomorfsed), *kui leidub selline bijektiivne kujutus $f : A_1 \longrightarrow A_2$, nii et $aR_1b \Longleftrightarrow (a)R_2f(b)$.*

Definitsioon 15. *Graafi* $G = (A, R)$ märgenduseks *nimetatakse funktsioonide paari* f *ja* g, *kus* $f : A \longrightarrow M$ *on tippude ja* $g : R \longrightarrow L$ *on kaarte märgendus.*

Märgendatud graafid $G_1 = (A_1, R_1)$ *ja* $G_2 = (A_2, R_2)$, *mille märgendused on vastavalt* (f_1, g_1) *ja* (f_2, g_2), *on* võrdsed, *kui leidub bijektiivne kujutus* $h : A_1 \longrightarrow A_2$, *selline, et*

- aR_1b *kehtib parajasti siis, kui* $h(a)R_2h(b)$, *s.t. graafid* G_1 *ja* G_2 *on võrdsed kui märgendamata graafid;*
- $f_1(a) = f_2(h(a))$;
- $g_1((a, b)) = g_2((h(a), h(b)))$.

Näide 4. Järgmised graafid on võrdsed, kuna leidub bijektiivne kujutus h, nii et $h(a) = 0$; $h(b) = 2$; $h(c) = 1$.



□

Definitsioon 16. *Tippude jada* $(a_0, \ldots, a_n)$, $n \geq 1$ *nimetatakse* teeks *pikkusega* n *tipust* a_0 *tippu* a_n, *kui iga* $a \leq i \leq n$ *jaoks leidub kaar tipust* a_{i-1} *tippu* a_1.

Tsükliks nimetatakse teed $(a_0, \ldots, a_n)$ graafis G, mille korral $a_0 = a_n$.

Graafi nimetatakse *tugevalt sidusaks*, kui iga kahe graafi tipu a ja b korral leidub tee tipust a tippu b.

Tipu astak sisendite suhtes on sellesse tippu sisenevate kaarte arv, tipu astak väljundite järgi on sellest tipust väljuvate kaarte arv.

Tsüklivabad orienteeritud graafid

Tsüklivaba graafi *baasiks* nimetatakse nende graafitippude hulka, mille astak sisendi järgi on 0. Tipud, mille astak väljundi järgi on 0, moodustavad tsüklivaba graafi *lehtede* e. *lõpptippude* (ka terminaalsete tippude) hulga.

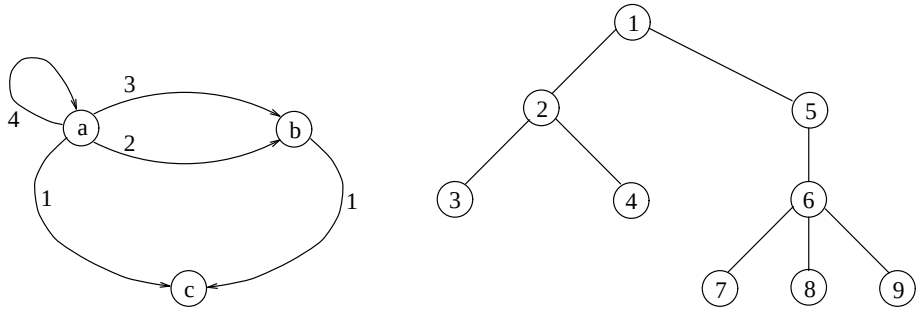
Definitsioon 17. Puuks *nimetatakse tsüklivaba graafi* $G = (A, R)$, *mille baas sisaldab ühe tipu (mida nimetatakse puu* juureks*), teiste tippude astak sisendi järgi on* 1 *ja iga tipu* $a \in A$ *jaoks leidub tee* $(r, c_1, c_2, \ldots, c_{n-1}, a)$.

Tipu a sügavuseks *puus nimetatakse tee* $(r, c_1, \ldots, c_{k-1}, a)$ *pikkust* k.

Tsüklivaba transitiivse orienteeritud graafi $G = (A, R)$ kaared määravad hulgal A osalise järjestuse.

Järjestusega graaf on paar $G = (A, R)$, kus A on tippude hulk ja $R = \{(R_a, \prec)\}_{a \in A}$ kaarte hulkade pere. Hulka R_a kuuluvad tipust a väljuvad kaared: $R_a = \{(a, b) \mid b \in A, aRb\}$, mis on lineaarselt järjestatud seose $\prec$ mõttes.

Näide 5. Järjestatud graafid

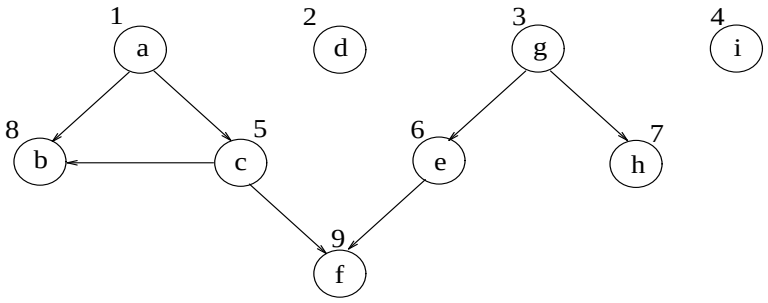


□

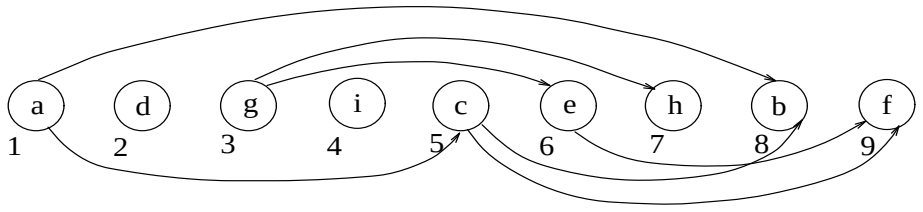
Topoloogilise sorteerimise ülesanne

Leida tsüklivaba graafi $G = (A, R)$ tippude selline märgendus $f : A \longrightarrow \mathbb{N}$, nii et $aRb \Longrightarrow f(a) < f(b)$. e. leida lineaarne järjestus hulgal A, mis sisaldaks endas järjestuse R.

Näide 6. Näide märgistusest, mille annab topoloogilise sorteerimise algoritm (vt. alg. 0.1).



Sama graaf "järjestatuna":



□

Topoloogilise sorteerimise ülesanne on algoritmiliselt lahenduv (vt. Algoritm. 0.1. raamatus A.Aho, J.Ullman. The Theory of Parsing, Translation and Compiling. Prentice Hall, 1972).

Puude esitamine

Märkus 1. Graafide esitamiseks arvutis kasutatakse intsidentsusmaatrikse ja mitmesuguseid viitstruktuure. Kõik nad sobivad ka puude esitamiseks.

Käesolevas vaatleme puude esitamist sulgavaldistena - ees-, kesk- ja lõppjärjekorras. See on erinevate topoloogiliste sorteerimiste rakendamine puude korral.

Puu T sulgavaldis koosneb puu tippude märgendist, ümarsulgudest ja komadest.

Definitsioon 18. *Järjestatud puu T eesjärjekord on avaldis* $\mathrm{lrep}(T)$*, mis on saadud järgmiste tingimuste kohaselt:*

1. *kui puu T juur on a, mille vahetud alampuud loetletuna vasakult paremale on $T_1, \ldots, T_k$, siis*
$$\mathrm{lrep}(T) = a(\mathrm{lrep}(T_1), \ldots, \mathrm{lrep}(T_k));$$
2. *kui a on puu T terminaalne tipp, siis*
$$\mathrm{lrep}(T) = a.$$

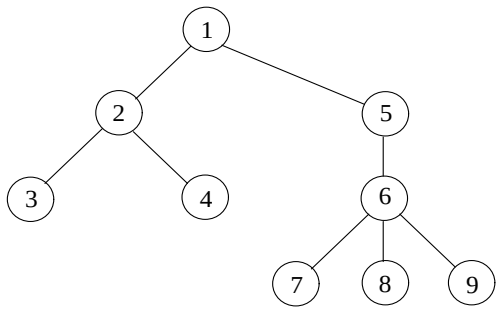
Definitsioon 19. *Järjestatud puu T keskjärjekord on avaldis* $\mathrm{mrep}(T)$*, mis on saadud vastavalt järgmistele tingimustele:*

1. *kui puu T juur on a, mille vahetud alampuud loetletuna vasakult paremale on $T_1, \ldots, T_k$, siis*
$$\mathrm{mrep}(T) = \mathrm{mrep}(T_1),\ a(\mathrm{mrep}(T_2), \ldots, \mathrm{mrep}(T_k));$$
2. *kui a on puu T terminaalne tipp, siis*
$$\mathrm{mrep}(T) = a.$$

Definitsioon 20. *Järjestatud puu T lõppjärjekord on avaldis* $\mathrm{rrep}(T)$*, kui*

1. *kui puu T juur on a, mille vahetud alampuud on loetuna vasakult paremale $T_1, \ldots, T_k$, siis*
$$\mathrm{rrep}(T) = (\mathrm{rrep}(T_1), \ldots, \mathrm{rrep}(T_k))a;$$
2. *kui puu T juur a on terminaalne tipp, siis* $\mathrm{rrep}(T) = a$.

Näide 7. Puu



eesjärjekord (vasakrekursiivne sulgavaldis) on

$$\mathrm{lrep}(T) = 1(2(3,4),\ 5(6(7,8,9)))$$

keskjärjekord:

$$\mathrm{mrep}(T) = 3, 2, (4),\ 1(7, 6(8,9), 5)$$

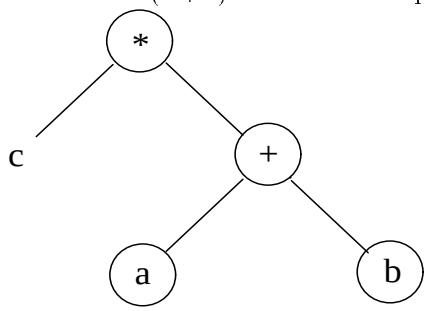
lõppjärjekord (paremrekursiivne sulgavaldis):

$$\mathrm{rrep}(T) = ((3,4), 2,\ ((7,8,9)6)5)1).$$

□

2.3.1. Programmi struktuuri esitamine puuna

Järjestatud puid kasutatakse arvutites programmide struktuuri esitamiseks. Näiteks mingis programmis esinev avaldis $c * (a + b)$ transleeritakse puuks



Selline järjestatud puu konstrueerimine ei ole programmi lähteteksti alusel kuigi keerukas. Puu esitab nii täitmisele tulevad operatsioonid kui ka nende teostamise järjekorra: puus allpool olevad operatsioonid tuleb täita varem.

Analoogselt aritmeetilise avaldisega saab puudena esitada kõiki põhilisi programmide juhtstruktuure nagu näiteks tingimuslik operaator, tsüklioperaator ja protseduurilause. Kuna puude kujutamine viitstruktuuridena on liiga mahukas, kasutatakse programmide struktuuri moodustamisel arvutis puude ees-, kesk- või lõppjärjekordi. Sageli võib niisugust esitust veelgi lühendada, jättes ära punktuatsioonisümbolid (komad ja sulud).

Ülaltoodud avaldis lõppjärjekorras omandaks kuju

$$c\ a\ b\ +\ *$$

ning tema esitamiseks arvuti mälus kuluks 5 mälupesa.

Lõppjärjekorras esitatud programmide täitmiseks sobib lihtne programm, mis kasutab ühte magasini (pinu). Olgu magasini kasutamise operatsioonid **push**(*element*) - elemendi lisamine magasini ja **pop** - funktsioon, mille väärtuseks on magasini tipu väärtus. Operatsioon **pop** eemaldab magasinist tipus oleva väärtuse.

Lõppjärjekorras esitatud programmi kasutamiseks olgu antud operatsioon **read**(*element*), mis skaneerib lõppjärjekorra järjekordse sümboli. Operatsiooni tulemus on **empty**, kui järjekorra kõik elemendid on skaneeritud. Iga skaneeritava elemendi korral on määratud ka funktsioon **range**(*element*), mis operandide korral on 0, operatsioonimärkide puhul aga võrdub tema argumentide arvuga. Näiteks

$$\mathbf{range}(\mathrm{a}) = \mathbf{range}(\mathrm{b}) = \mathbf{range}(\mathrm{c}) = 0$$
$$\mathbf{range}(+) = \mathbf{range}(*) = 2$$

n-kohalise operatsiooni T rakendamine argumentidele $x_1, x_2, \ldots, x_n$ olgu kirjeldatav primitiiviga

$$\mathbf{apply}(T, x_1, x_2, \ldots, x_n)$$

Toodud tähistuste korral saab lõppjärjekorras esitatud programmi täita järgmise protseduuri abil:

```
procedure execute;
    begin
        while read(element) ≠ empty do
            begin
                if range(element) = 0 then push(element);
                if range(element) = 1 then
                    push (apply(element), pop));
                if range(element) = 2 then
                    push (apply(element, pop, pop));
            end;
        print (pop);
end.
```

2.4. Matemaatilise loogika põhimõisted

2.4.1. Tõestuskäigu formaliseerimine

Lisaks hulga- ja graafiteooria mõistetele vajatakse teoreetilises informaatikas ka matemaatikas kasutatavaid tõestusmeetodeid. Põhjalikult käsitletakse järelduste tegemise tehnikat kursuses "loogika arvutiteaduses", siinkohal esitame vaid lühiülevaate tõestusvõtetest, mida kursuses edaspidi rakendatakse.

Jämedas laastus võib tuletusmeetodid jagada kahte klassi: deduktiivsed (üldiste seaduste alusel üksikobjektide omaduste üle otsustamine) ja induktiivsed (üksikobjektide omaduste üldistamine objektide klassidele) järeldusmeetodid. Mõlemal korral saab järeldussamme esitada tõestuspuuna, kus üleminekud eelduselt järeldusele eraldatakse rõhtjoonega. Näiteks kui kehtivad väited $\mathcal{L}$ ja $\mathcal{L} \longrightarrow \mathcal{M}$ (see on liitlause, mida tuleks lugeda "Kui kehtib väide $\mathcal{L}$, siis kehtib ka väide $\mathcal{M}$"), siis kehtib väide $\mathcal{M}$. Sellise arutelu võib üles kirjutada skeemina

$$\frac{\mathcal{L} \quad \mathcal{L} \longrightarrow \mathcal{M}}{\mathcal{M}}, \tag{1}$$

kus joone kohal on eeldused, joone all aga järeldus. Toodud skeemi (1) tuleb vaadelda kui reeglit, mida võib kasutada igas tõestuses, asendades muutujad $\mathcal{L}$ ja $\mathcal{M}$ antud olukorras sobivate väidetega. Skeem (1) ongi deduktiivse tuletuse peamine reegel, mida Aristotelese järgi kutsutakse ladinapärase nimetusega *modus ponens*. Sama skeem võiks olla esitatud ka kujul

$$\frac{\mathcal{L} \quad \frac{\mathcal{L}}{\mathcal{M}}}{\mathcal{M}}.$$

Modus ponens on deduktsiooniteooria peamine tuletusreegel, kuigi mitte ainus. Toome siinkohal veel mõned enamkasutatavatest loogikareeglitest.

$$\frac{}{\mathcal{L} \longrightarrow \mathcal{L}}, \tag{2}$$

mis tähendab, et ilma eeltingimusteta kehtib väide "Kui väide $\mathcal{L}$ on tõene, siis $\mathcal{L}$ on tõene";

$$\frac{\neg\mathcal{L} \longrightarrow \mathcal{L}}{\otimes}, \tag{3}$$

mis tähendab, et väitest "Kui ei kehti väide $\mathcal{L}$ (tähistus $\neg\mathcal{L}$), siis kehtib väide $\mathcal{L}$" järeldub vastuolu (tähis $\otimes$);

$$\frac{\mathcal{L} \longrightarrow \mathcal{M}}{\longrightarrow \mathcal{M} \vee \neg\mathcal{L}}, \tag{4}$$

mis tähendab, et lausest "Kui kehtib väide $\mathcal{L}$, siis kehtib ka väide $\mathcal{M}$" järeldub lause "Ilma eeltingimusteta kehtib väide $\mathcal{M}$ või ei kehti väide $\mathcal{L}$";

$$\frac{\mathcal{L}_1 \longrightarrow \mathcal{M}_1 \quad \mathcal{L}_2 \longrightarrow \mathcal{M}_2}{\mathcal{L}_1 \vee \mathcal{L}_2 \longrightarrow \mathcal{M}_1 \vee \mathcal{M}_2}, \tag{5}$$

mis näitab, kuidas kahest tõestastud väitest ($\mathcal{L}_1 \longrightarrow \mathcal{M}_1$ ja $\mathcal{L}_2 \longrightarrow \mathcal{M}_2$) võib järeldada keerulisema.

Deduktiivne tõestamine on sellise tuletussammude jada leidmine, mis lõpeb tõestatava lausega (teoreemiga) ning jada iga element on kas aksioom (antud situatsioonis tõestamist mittevajav lause) või saadav temast jadas eespool olevatest lausetest tuletusreeglite (1) või (5) abil. Jadana esitatud tõestuse võib, nagu juba eespool mainitud, kujutada ka puuna, mille juureks on teoreem ja kaared vastavad tuletusreeglite rakendamisele. Puu lehed peavad sel juhul vastama aksioomidele.

Aksioomid võib esitada samal kujul kui tuletusreeglid või kirjutada implikatiivsete lausetena. Näiteks ekvivalentsiseose definitsiooni (Def. 21) saab esitada kujul:

Definitsioon 21. *Seost $R \subseteq A \times A$ nimetatakse ekvivalentsiseoseks, kui ta rahuldab järgmisi tingimusi:*

$$\frac{}{aRa} \qquad \textit{(refleksiivsus)};$$

$$\frac{aRb}{bRa} \qquad \textit{(sümmeetrilisus)};$$

$$\frac{aRb \quad bRc}{aRc} \quad \textit{(transitiivsus)}.$$

Implikatiivsel kujul oleksid viimased tingimused kujul:

$$\longrightarrow aRa \qquad \textit{(refleksiivsus)};$$

$$aRb \longrightarrow bRa \qquad \textit{(sümmeetrilisus)};$$

$$(aRb)\&(bRc) \longrightarrow aRc \quad \textit{(transitiivsus)}.$$

Elemendi $a \in A$ ekvivalentsiklassi definitsiooni võib esitada tingimustega

$$z \in [a] \longrightarrow aRz$$

ja

$$aRz \longrightarrow z \in [a],$$

kus R on ekvivalentsiseos hulgal A.

Samuti saab loogikatähistuste abil esitada näiteks hulgateooria tehete definitsioone:

- alamhulk
$$\frac{z \in A \longrightarrow z \in B}{A \subseteq B};$$
- hulkade võrdsus
$$\frac{A \subseteq B \quad B \subseteq A}{A = B};$$
- hulkade lõikumine
$$\frac{A \cap B \neq \emptyset}{\exists y(y \in A \& y \in B)}.$$

2.4.2. Deduktiivse tõestuse tüübid

Ülalesitatud tuletusreeglite põhjal võib tõestada täiendavaid tuletusreegleid, mille kasutamise korral tõestuskäik lihtsustub. Tõestus on enamasti lühem ja ülevaatlikum, kuid uued, põhireeglitest tulenevad reeglid ei võimalda tõestada suuremat hulka teoreeme kui eelpool toodud põhireeglid. Sellised täiendavad tuletusreeglid vastavad erinevatele tõestusviisidele, nagu näiteks tõestamine lemmade abil ja vastuväiteline tõestamine. Kasutades näitena teoreemi 1 tõestust, esitame järgnevas lühidalt tõestusvõtteid, mida kasutatakse edaspidi ka teoreetilise informaatika kursuses.

Väite lihtsustamine. Olgu tarvis tõestada, et eelduste $\mathcal{L}$ korral kehtib lause: "Kui kehtib $\mathcal{M}$, siis kehtib $\mathcal{N}$". See on kõige tavalisem matemaatilise teoreemi üldkuju. Teoreemi vahetu tõestamise asemel tõestatakse, et kui lisaks lausele $\mathcal{L}$ eeldada ka lause $\mathcal{M}$ kehtivust, kehtib lause $\mathcal{N}$. Formaalselt tõestuse

$$\frac{\begin{array}{c}\underline{\mathcal{L}}\\ \vdots\end{array}}{\mathcal{M} \longrightarrow \mathcal{N}} \tag{6}$$

asemel koostatakse tõestus

$$\frac{\underline{\mathcal{L}} \quad \underline{\mathcal{M}}}{\begin{array}{cc}\vdots & \vdots\\ \mathcal{N}\end{array}}. \tag{7}$$

Seega teoreem $\mathcal{M} \longrightarrow \mathcal{N}$ asendatakse lihtsama teoreemiga $\mathcal{N}$, mis on lihtsama struktuuriga väide ning mida enamasti on ka kergem tõestada. Asjaolu, et tõestuse (7) olemasolust järeldub tõestuse (6) olemasolu, tunti loogikas juba ammu, ehkki vastava asenduse korrektsuse tõestas prantsuse loogik J. Herbrand alles 1930.a.

Teoreemi 1 tõestuses kasutatakse väite lihtsustamise võtet kohe algul, kus eelduse aRb asemel tõestatakse, et $[a] \subseteq [b]$. Selle asemel, et näidata, et väitest $z \in [a]$ järeldub väide $z \in [b]$ e. formaalselt, tõestuse

$$\frac{\begin{array}{c}\underline{aRb}\\ \vdots\end{array}}{z \in [a] \longrightarrow z \in [b]}$$

asemel tehakse lisaeeldus $z \in [a]$ (eestikeelse tekstina on väljendatud lauses: "Valime klassist $[a]$ vabalt ühe elemendi z") ning näidatakse, et $z \in [b]$. Täpsemalt, tõestuses esineb lõik:

> "Siis kehtivad bRa (ekvivalentsiseose sümmeetria tõttu) ja aRz, millest ekvivalentsiseose transitiivsuse põhjal järeldub, et bRz. Seega $z \in [b]$ ..."

Viimane lõik on kujutatav formaalselt järgmise tõestuspuuna:

$$\frac{\dfrac{\dfrac{aRb}{bRa} \qquad \dfrac{z\in[a]}{aRz}}{bRz}}{z\in[b]}.$$

Vastavalt Herbrand'i avastusele loogikas on kehtiv siis ka tuletus

$$\frac{\dfrac{\dfrac{aRb}{z\in[a]\longrightarrow z\in[b]}}{[a]\subseteq[b]}}{} \tag{8}$$

Viimane järeldus toodud puus on tehtud alamhulga definitsiooni põhjal.

Edasi on teoreemi 1 tõestuses väidetud, et seos $[b] \subseteq [a]$ tõestatakse analoogiliselt. Formaalselt tähendab see asjaolu, et tõestuspuu

$$\frac{\dfrac{aRb \qquad \dfrac{z\in[b]}{bRz}}{aRc}}{z\in[a]}$$

olemasolust järeldub tõestuse

$$\dfrac{\dfrac{\overset{aRb}{\vdots}}{z\in[b]\longrightarrow z\in[a]}}{[b]\subseteq[a]} \tag{9}$$

olemasolu. Tõestused (8) ja (9) on aga lihtne hulkade võrdsuse definitsiooni abil kokku võtta tõestuseks

$$\dfrac{\dfrac{\overset{aRb}{\vdots}}{[a]\in[b]} \quad \dfrac{\overset{aRb}{\vdots}}{[b]\in[a]}}{[a[=[b]},$$

mis omakorda on samaväärne järeldusega

$$\dfrac{\overset{aRb}{\vdots}}{aRb\longrightarrow[a]=[b]}.$$

Vastuväiteline tõestus. Mõnikord on eelduste $\mathcal{L}$ korral väite $\mathcal{M}$ tõestus keeruline. Klassikalise loogika kasutamisel võib siis püüda näidata, et eeldustest $\mathcal{L}$ ja $\neg\mathcal{M}$ (väite $\mathcal{M}$ vastand) järeldub vastuolu. Seega teoreemi $\mathcal{M}$ tõestus

$$\dfrac{\underline{\mathcal{L}}\\ \vdots \\ \mathcal{M}}$$

järeldatakse tõestusest

$$\dfrac{\underline{\mathcal{L}} \quad \underline{\neg\mathcal{M}}\\ \vdots \qquad \vdots \\ \otimes}.$$

Märkus 2. Programmide konstrueerimisel ei kasutata "klassikalist" Aristotelese loogikat ning sel korral ei ole vastuväiteline tõestamine lubatud. Lähemalt käsitleme seda probleemi kursuses "loogika arvutiteaduses".

Vastuväitelist tõestust kasutatakse teoreemi 1 tõestuse teises osas, kus näidatakse, et seose aRb mittekehtimise korral elementide a ja b ekvivalentsiklassid ei lõiku. Formaalselt koostatakse tõestus valemile

$$\neg(aRb)\longrightarrow[a]\cap[b]=\emptyset,$$

mis on samaväärne formaalse tuletusega

$$\dfrac{\dfrac{\overset{\neg(aRb)}{\vdots}}{[a]\cap[b]=\emptyset}}{}.$$

Teoreemi 1 tõestuse teksti analüüs näitab, et tõestus on vastuväiteline:

$$\dfrac{\neg(aRb) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a]\cap[b]\neq\emptyset}{\exists y(y\in[a]\&y\in[b])}}{\exists y(aRy\&bRy)}}{\exists y(aRy\&yRb)}}{\exists y(aRb)}}{aRb}}{\otimes}$$

Lemmade abil tõestamine. Teoreemi püütakse tõestada mitmel (lihtsamal) erijuhul (erijuhte nimetatakse lemmadeks). Kui kõik võimalikud erijuhud on käsitletud, võime lugeda ka kogu teoreemi tõestatuks. Formaalselt võib lemmade abil tõestamisel kasutada tuletusreeglit

$$\dfrac{\mathcal{L}\longrightarrow\mathcal{L}_1\vee\ \ldots\ \vee\mathcal{L}_n \qquad \mathcal{L}_1\longrightarrow\mathcal{M}\ \ \ldots\ \ \mathcal{L}_n\longrightarrow\mathcal{M}}{\mathcal{L}\longrightarrow\mathcal{M}},$$

kus teoreemi $\mathcal{L} \longrightarrow \mathcal{M}$ tõestus kasutab lemmasid (kasutatakse ka terminit *abiteoreeme*) $\mathcal{L}_1 \longrightarrow \mathcal{M}$, ... $\mathcal{L}_n \longrightarrow \mathcal{M}$.

Pikemate tõestuste korral vormistataksegi lemmade tõestused eraldi ning teoreemi tõestuses viidatakse neile tõestustele. Lihtsamatel juhtudel võib matemaatilises tekstis otsene viide lemmade abil tõestamisele isegi puududa ning seda tuleb välja lugeda "ridade vahelt". Näiteks teoreemi 1 korral tuleb lemmade olemasolu "välja lugeda" lausekonstruktsioonist:

"Vaatleme algul juhtu, kus kehtib aRb.
...
Juhul kui $\neg(aRb)$ ..."

Tõepoolest, tõestatakse ju algul lemma

$$aRb \longrightarrow [a] = [b]$$

ja seejärel lemma

$$\neg(aRb) \longrightarrow [a] \cap [b] = \emptyset,$$

mis mõlemad on erijuhud teoreemist 1. Kuna eelduste paar aRb ja $\neg(aRb)$ katavad kõik võimalikud juhud (s.t. et tingimata on kehtiv valem $aRb \vee \neg(aRb)$), siis tuletusreegli (5) abil on tõestatav teoreem 1.

$$\longrightarrow ([a] = [b]) \bigvee ([a] \cap [b] = \emptyset).$$

Matemaatilistes tekstides peetakse aga viimast mõttekäiku "niivõrd loogiliseks", et seda ei peeta vajalikuks isegi otseselt kirja panna. seepärast ongi harjumata inimesel vahel raske matemaatilistest tõestustest aru saada.

Esitame siinkohal tuletuspuu, mis viib täielikult lõpule teoreemi 1 tõestuse, esitades ka need järeldussammud, mis tekstis vahetult kirjas ei ole:

$$\dfrac{\dfrac{\dfrac{}{aRb \longrightarrow aRb}}{\longrightarrow (aRb) \vee \neg(aRb)} \qquad \dfrac{aRb \longrightarrow [a] = [b] \quad \neg(aRb) \longrightarrow [a] \cap [b] = \emptyset}{(aRb) \vee \neg(aRb) \longrightarrow ([a] = [b] \vee ([a] \cap [b] = \emptyset)}}{\longrightarrow ([a] = [b]) \vee ([a] \cap [b] = \emptyset)}$$

2.4.3. Induktiivne tõestamine

Teoreetilises informaatikas kasutatakse tihti ka matemaatikas tuntud transfiniitset induktsiooni, mida rakendatakse siis, kui tõestatav väide A on mingi järjestatud hulga E kõigi elementide kohta. Sümbol $A(z)$ tährendab järgnevas, et tõestamist vajav väide A on kehtiv vähemalt elemendi $z \in E$ korral. Sel juhul võib induktiivse meetodi sõnastada kujul:

Transfiniitne induktsioon: Olgu hulgal E määratud lineaarne järjestusseos $<$. Kui iga $z \in E$ korral järeldub väite $A(z)$ tõesus väidete $A(y)$ tõesusest kõikide $y < z$ korral, on väide $A(x)$ tõene iga $x \in E$ jaoks.

Seega transfiniitse induktsiooni tuletusreegel:

$$\frac{\forall z(\forall y(y < z \longrightarrow (A(y) \longrightarrow A(z)))}{\forall x A(x)}. \tag{10}$$

Transfiniitse induktsiooni erijuhtu, kus alushulgaks E on naturaalarvude hulk $\mathbb{N}$, nimetatakse *matemaatiliseks induktsiooniks*.

Deduktsiooniteoorias näidatakse, et matemaatilise induktsiooni reegel (10) on samaväärne tuletusreegliga

$$\frac{A(0) \qquad \forall x(Ax) \longrightarrow A(x+1))}{\forall x A(x)}.$$

Sagedasti vormistataksegi induktiivne tõestus kaheosalisena:

– *induktsiooni baas*: tingimuse $A(0)$ kontroll;
– *induktsiooni samm*: tõestus, et valemist $A(x)$ järeldub $A(x+1)$. Valemit $A(x)$ nimetatakse seejuures induktsiooni eelduseks.

Näide 8. Valem $S(n) = 1+3+5+\ldots 2n-1 = n^2$ on tuletatav iga $n = 1, 2\ldots$ korral.
Baas $n = 1$ korral $S(1) = 1 = 1^2$.
Samm. Eeldame, et suvalise $n \in \mathbb{N}$ jaoks kehtib $S(n)$.
Väite $S(n+1)$ korral

$$1+3+5 \quad 2n-1+2n+1 = n^2+2n+1, \tag{11}$$

kuna eelduse põhjal kehtib

$$S(n) = 1+3+\ldots+2n-1 = n^2.$$

Seosest (11) ja valemist $(n+1)^2 = n^2+2n+1$ järeldub, et kehtib ka väide

$$S(n+1) \equiv 1+3+5+\ldots+2n+1 = (n+1)^2$$

. □

3. Arvutites kasutatavad keeled

Keel on üks arvutiõpetuse põhimõistetest. Arvuti töötleb fikseeritud märgisüsteemis (keeles) esitatud informatsiooni. Ka ülesannete lahendamiseks kasutatavad algoritmid esitatakse arvuti jaoks teatud tehiskeeles, mida arvuti suudab interpreteerida. Oluliseima osa arvutis kasutatavatest keeltest moodustavad programmeerimiskeeled.

3.1. Programmeerimiskeeled

Programmeerimiskeel on tähistuste ja reeglite süsteem algoritmide esitamiseks arvutile.

Arvuti 40-aastase ajaloo jooksul on loodud tuhandeid programmeerimiskeeli, millest enamkasutatavad on kümmekond. Programmeerimiskeel on inimese ja masina vaheline suhtlemiskeel. Selle järgi, kuivõrd keel on inimesele sobival kujul, võib programmeerimiskeeli liigitada kolme klassi:

- masinkeeled;
- algoritmilised keeled e. kõrgtaseme keeled;
- teadmiste esitamise keeled e. spetsifitseerimiskeeled.

3.1.1. Masinkeeled

Masinkeelte näidetena võib tuua:

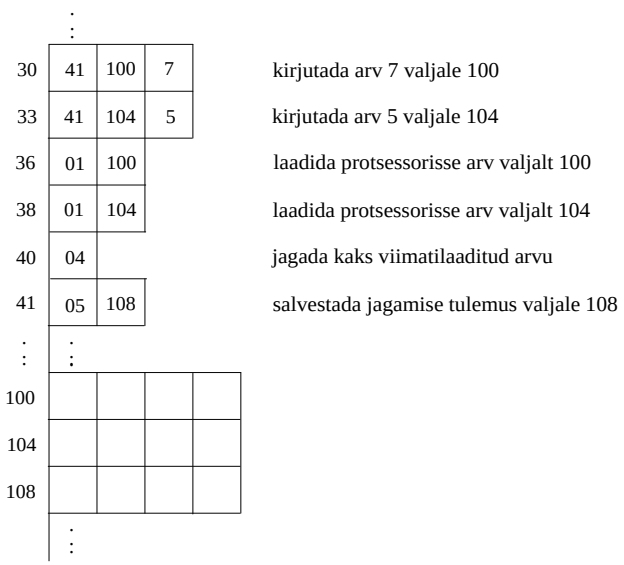
- masinkood - konkreetse arvuti käskude jada kodeerituna 2-nd-, 8-nd-, 16-ndsüsteemis arvudena;
- autokood - konkreetsele arvutile orienteeritud programmeerimiskeel.

Näide 9. Lahendada võrrand $ax + b = 0$, eeldusel et $a = 5$, $b = 7$.

□

Programm masinkoodis sõltub tema asukohast mälus. Järgnev programmilõik on kirjutatud eeldusel, et ta asub põhimälus alates baidist nr. 30, muutujate a, b ja x jaoks mälu eraldatud neljabaidiste väljadega vastavalt algusaadressidega 100, 104 ja 108.

Iga käsk koosneb mälukoodist ja sellele järgnevate operandide aadressidest.



Sama programm esitatuna autokoodis:

```
MVC   A '5'
MVC   B '7'
LOAD  A
LOAD  B
DIV
STORE X
```

Autokoodis on käsukoodid esitatud mnemooniliste koodidena - akronüümidena käsu tähendust väljendavast sõnaühendist. Näiteks käsukood *MVC* tuleneb inglisekeelsetest sõnadest Move Value to Cell (paigutada väärtus pessa). Mäluvälju tähistatakse neile vastavate muutujate nimedega, enamasti ei sõltu autokoodis kirjutatud programm tema asukohast mälus.

Autokoodi käsud on üksüheses vastavuses protsessori käskudega. Seepärast on iga protsessori jaoks oma autokood.

Enne täitmist transleeritakse autokoodis esitatav programm masinkoodi. Seda suhteliselt lihtsat mnemoonilistes koodides kirjutatud programmi teisendamist vastavateks protsessori käskudeks nimetatakse assembleerimiseks (ingl. k. assemble - koguma, kokku kutsuma, monteerima). Autokoodi nimetatakse ka assemblerkeeleks.

Masinkeeltes kirjutatakse iga uue arvuti esmane süsteemne tarkvara, näiteks mõne kõrgkeele translaator- ja operatsioonisüsteem.

3.1.2. Kõrgtaseme keeled e. algoritmilised keeled

Algoritmilised keeled on mõeldud arvutist sõltumatute arvutusprotsesside kirjeldamiseks.

Algoritmilistes keeltes ...

... esitatakse aritmeetilised arvutused algebraliste avaldistena;
... kasutatakse spetsiaalseid lausekonstruktsioone peamiste algoritmiliste juhtstruktuuride (seeria, korduse ja hargnemise) esitamiseks;
... on võimalused sisendi-väljundi kirjeldamiseks;
... saab erinevate objektide omadusi esitada kasutades erinevaid andmetüüpe (arvud, massiivid, hulgad, kirjed, puud, graafid jne.).

Algoritmiliste keelte näiteid:

- Fortan (FORmula Translator) - 1957
- Cobol (COmmon Business Oriented Language) - 1958-1960
- Algol-60 (ALGOrithmic Language) - 1957-1960
- Basic (Beginner's All-purpose Symbolic Instruction Code) - 1965
- PL/I (Programming Language One) - 1963-1966
- Pascal - 1971
- Lisp (LISt Processing) - 1957-1962
- Ada (Ada Augusta Lovelance'i auks) - 1977-1983
- C - 1985-1988

Näide 10. lahendada võrrand $ax + b = 0$.

FORTRAN:

```
 10 READ a, b
 20 IF (a)30, 70, 30
 30 x = -b/a
 40 PRINT 50, x
 50 FORMAT ('Vastus: x =', F6.2)
 60 GOTO 90
 70 PRINT 80
 80 FORMAT ('Võrrandil puudub lahend')
 90 STOP
100 END
```

□

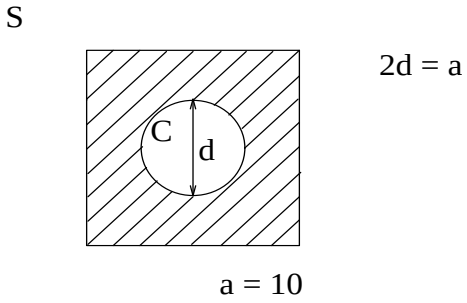
3.1.3. Teadmiste esitamise keeled

Teadmiste esitamise keeled (ka *teadmuskeeled*) on ette nähtud ülesannete spetsifitseerimiseks, ülesande lahendusalgoritmi koostab arvuti lähtudes ülesande kirjeldusest.

Keelte näited:

PROLOG (PROgramming in LOGic)
CLU
UTOPIST
...

Näide 11. Leida viirutatud kujundi pindala



UTOPIST

let $Ruut : (a, S, p, d :$ **numeric**;
relations
$a * a = S$;
$p = 4 * a$;
$d^2 = 2 * a^2$);
$Ring : (r, d, S, p :$ **numeric**;
relations
$p = 6.28 * r$;
$S = 3.14 * r^2$;
$d = 2 * r$);
$Kujund : (S : Ruut$;
$C : Ring\ d = 2.a/2$;
$x :$ **numeric**;
relations
$x = S.S - C.S$);
actions
on $Kujund$ **compute** x **from** $S.a = 10$;
end;

□

Kaasajal on mitmed spetsifitseerimiskeeled visuaalsed. See tähendab, et ülesande püstitamiseks arvutile tuleb spetsiaalses graafikaaknas hiire abil joonistada ülesande tingimusi esitav joonis. Eelmises näites püstitatud ülesande võib sel juhul spetsifitseerida näite algul toodud joonise abil ning arvuti teisendab selle ülesande püstituse ise tekstilisele kujule.

Rääkides keelest, oleme harjunud endale ette kujutama teksti, tegelikult võib aga visuaalsete kujundite süsteemi vaadelda kui keelt. Arvutikeelena võib kasutada mistahes märgisüsteemi, mis on tõlgitav (transleeritav) arvuti masinkeelde. Teatud üldistuse korral võime kõiki arvutiprogramme käsitleda translaatoritena, mis tõlgivad sisendandmed neile vastavateks väljundandmeteks.

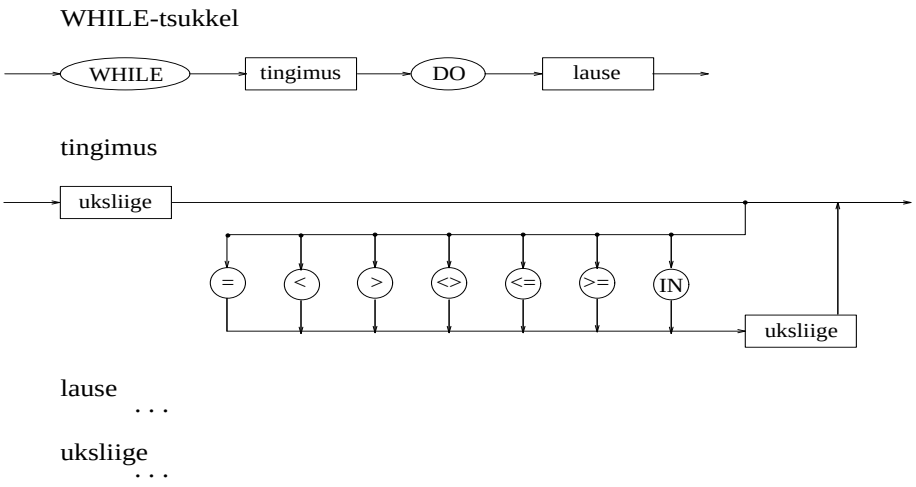
3.2. Keelte formaalne defineerimine

Arvutil on võimalik lahendada vaid neid ülesandeid, mille lahendusalgoritm on esitatav matemaatilise formalismi abil. Seega peab ka arvutil kasutatavate keelte ning nende transleerimisprotsessi kirjeldamiseks leiduma formaalne esitus.

Kõik tuntud meetodid keelte formaliseerimiseks vaatlevad lahus süntaksi ja semantika kirjeldamist. Keele *süntaks* väljendab lausete sisemist struktuuri sõltumata lause tähendusest, *semantika* esitab lause tähenduse (reeglina mõnes teises keeles). Vahe keele süntaksi ja semantika vahel pole siiski range, sõltuvalt käsitlusest võib teatud keele süntaktilisi omadusi vaadelda semantilistena ja vastupidi. Süntaks ja semantika pole pelgalt arvutis kasutatavate tehiskeelte erinevad aspektid. Nad iseloomustavad kõiki märgisüsteeme, kaasaarvatud loomulik inimestevaheline suhtlemiskeel. Näiteks tuntud fraseologism "tule eile meile" on eesti keele süntaksireeglite kohaselt korrektne lause, semantiliselt aga mõttetu, kuna esitab ilmselt võimatut situatsiooni. Siiski ei saa ka sellist otsustust teha üheselt, sest teatud kontekstis, kus eesmärgiks ongi esitada mõttetust kui niisugust, omandab antud lause tähenduse.

Nagu juba toodud lihtsast näitest järeldub, on keele semantika esitamine võrratult keerulisem (ja suhtelisem) kui süntaksi määratlemine. Nii on ka keele süntaksi formaliseerimise meetodid hästi tuntud, samal ajal kui semantika jaoks sobiv efektiivne formalism puudub. Keele semantika esitatakse enamasti kaudselt, lausete tõlke kaudu mõnda teise keelde või lausete "korrektsuse" tõestamise kaudu.

Programmeerimiskeelte õpikute ja kasutamisjuhendite kaudu tunneb lugeja kindlasti mitmeid formalisme süntaksi esitamiseks. Toome siinkohal näitena süntaksidiagrammid, nn. Wirthi skeemid, mida esmakordselt kasutati programmeerimiskeele Pascal defineerimisel. Süntaksidiagramm on graaf, mis esitab keele lause või selle osa (fraasi) süntaktilise struktuuri. Grammatiliselt korrektse lause konstrueerimiseks tuleb graafi läbida mööda kaari (mitme lubatud tee korral võib valida suvaliselt ühe alternatiividest). Teele jäävates ringikujulistes või ümarate nurkadega tippudes olevad väärtused tuleb kirjutada lausesse vahetult, ristkülikukujulise tipu tähenduse selgitab diagramm, mille nimetus on skeemil ristküliku sees. Näiteks Pascali tsüklilause esitavad diagrammid



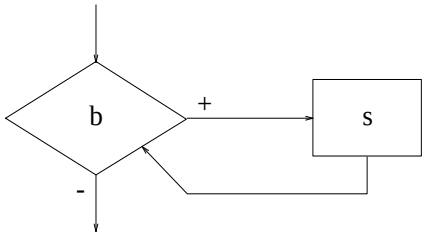
Toodud skeemide täpsel järgimisel on võimalik koostada Pascal-keele tsüklilauseid, kuid seejuures ei selgu, milline on kirjutatu tähendus, mida arvuti vastava lause mõjul teeks. Enamasti esitatakse lause semantika tekstina loomulikus keeles. Näiteks WHILE-tsükli kohta võib olla kirjutatud:

> "WHILE-tsükli täitmine algab tingimuse väärtuse arvutamisega. Kui see on tõene, täidetakse võtmesõna DO järel olev lause (tsükli keha) ning arvutatakse uuesti tingimuse väärtus ja tõese väärtuse korral täidetakse uuesti tsükli keha. Kirjeldatud viisil jätkatakse seni, kuni tingimus muutub vääraks. Kui tingimus on väär juba esimese kontrolli korral, on WHILE-tsükli tulemus samaväärne tühilause täitmisega".

Selline selgitus annab küll inimesele ettekujutuse, mida tsüklioperaator tähendab, kuid ei ole piisavalt range, et seda üheselt mõista ning realiseerida vastav transleerimisprotseduur, mis teisendaks tsüklilause masinkeelde. Mõnevõrra täpsema semantika definitsiooni võib anda lausele

$$\text{WHILE } b \text{ DO } s,$$

kasutades plokkskeemi:



Süntaksi esitamisel on populaarsed ka Bacus-Nauri valemid, mis esitavad reeglid korrektsete keele lausete koostamiseks fraasidest. Näiteks Pascali tsüklilause võib anda valemitega:

WHILE-tsükkel ::= 'WHILE' tingimus 'DO' lause
tingimus ::= üksliige
tingimus ::= üksliige tehe üksliige
tehe ::= $'='$
tehe ::= $'<'$
tehe ::= $'>'$
tehe ::= $'<>'$
tehe ::= $'<='$
tehe ::= $'>='$
tehe ::= 'IN'
üksliige ::= ...
lause ::= ...

Lühema kirjapildi saamiseks võib ühesuguse vasaku poolega reeglid asendada ühe reegliga, mille paremal poolel on toodud võimalike alternatiivide loetelu (eraldajaks alternatiivide vahel on püstkriips |). Näiteks

$$\text{tehe} ::= '=' \mid '<' \mid '>' \mid '<>' \mid '<=' \mid '>=' \mid \text{'IN'}$$

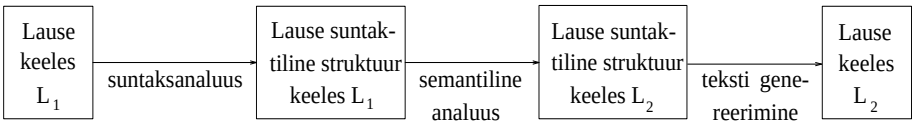
Lihtne on veenduda, et Bacuse-Nauri valemid on samaväärsed Wirthi skeemidega, võimaldades esitada ainult süntaksi. Semantika tuleb defineerida jällegi näiteks vastava lause tõlkimisprotseduuri (transleerimisalgoritmi) kaudu. Transleerimisalgoritmis viidatakse loomulikult reegli paremas pooles kasutatud fraaside nimetustele (näiteks tingimus, lause jne.).

Toodud näited illustreerivad programmeerimiskeelte õpikutes kasutatavaid formalisme. Keelte realiseerimiseks arvutis vajatakse enam matemaatilisemat aparatuuri, mida uuritakse formaalsete keelte teooria raames.

Kõiki arvutiprogramme võime käsitleda translaatoritena, mis tõlgivad programmi sisendandmed neile vastavateks väljunditeks. Olgu sisendandmete esitamise keel L_1 ja tulemuste keel L_2. Siis võib programmi tööd ("transleerimisprotsessi") vaadelda kujututsena

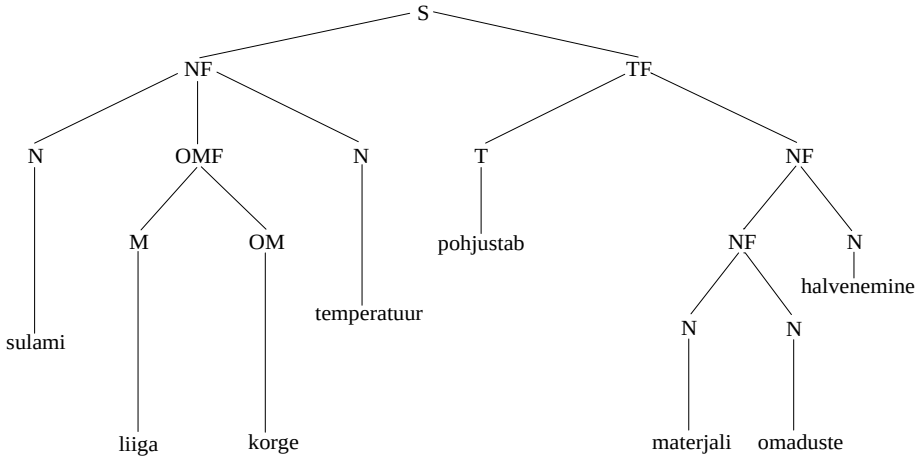
$$T_r : L_1 \longrightarrow L_2.$$

Kujutuse T_r arvutil realiseerimise traditsiooniline skeem sisaldab kolm etappi:



Lause süntaktiline struktuur tähendab siin tema koosseisu kuuluvate fraaside hierarhiat vastavalt antud keele grammatikale.

Näide 12. Lause "Sulami liiga kõrge temperatuur põhjustab materjali omaduste halvenemise" süntaktiline struktuur vastavalt eesti keele reeglitele on esitatav järjestatud puuna:



Toodud puus kasutatakse sõnaühendite (fraaside) tähistamiseks järgmisi sümboleid:
S – lause
NF – nimisõnafraas
TF – tegusõnafraas e. öeldisefraas
OMF – omadussõnafraas
N – nimisõna
T – tegusõna
OM – omadussõna
M – määrsõna

□

Formaalsete keelte teooria uurib keele süntaksanalüüsi ja teksti genereerimise algoritmilisi meetodeid.

3.3. Keeled kui stringihulgad

Olgu järgnevas sümboliga Σ tähistatud teatud tähestik (lõplik tekstimärkide hulk). Sümbolitega $a, b, c, \ldots$ märgime tähestiku Σ tähti ning sümbolitega $x, y, z, \ldots$ stringe tähestikus Σ (vt. järgmine definitsioon).

Definitsioon 22. *String tähestikus Σ on määratud järgmiste tingimustega:*

1. *ϵ - tühi string (kuulub iga tähestiku stringide hulka);*
2. *ax on string, kui x on tähestiku Σ string ja sümbol $a \in \Sigma$;*
3. *y on tähestiku Σ string vaid siis, kui ta rahuldb tingimusi 1 ja 2.*

Tähestiku Σ kõigi stringide hulka tähistatakse sümboliga Σ^*

Stringi x pöördstringiks x^R nimetatakse sümbolite jada, mis saadakse stringi x kirjutamisel tagantpoolt ettepoole.

Stringide $x \in \Sigma^*$ ja $y \in \Sigma^*$ järjestkirjutamise operatsiooni nimetatakse *konkatenatsioonioperatsiooniks*. Stringide x ja y konkatenatsiooni tähistatakse xy. Olgu näiteks $x = a_1a_2 \ldots a_m$ ja $y = b_1b_2 \ldots b_n$, siis

$$xy = a_1a_2 \ldots a_mb_1b_2 \ldots b_n.$$

Olulisemad konkatenatsioonioperatsiooni omadused:

$$(xy)z = x(yz)$$

ja

$$\epsilon x = x\epsilon = x.$$

Olgu antud string $s = xyz$. Sellise stringi osade jaoks kasutatakse järgmisi nimetusi: x, y ja z - *alamstringid*, x - *prefiks* ja z - *sufiks*.

Definitsioon 23. *Keel on alamhulk $\mathcal{L} \subseteq \Sigma^*$.*

Definitsioon 24. *Keelte $\mathcal{L}_1 \subseteq \Sigma_1^*$ ja $\mathcal{L}_2 \subseteq \Sigma_2^*$ konkatenatsiooniks nimetatakse keelt $\mathcal{L} \subseteq (\Sigma_1 \cup \Sigma_2)^*$, kus*

$$\mathcal{L} = \{xy \mid x \in \mathcal{L}_1,\ y \in \mathcal{L}_2\}.$$

Definitsioon 25. *Keele $\mathcal{L}$ iteratsiooniks nimetatakse hulka*

$$\mathcal{L}^* = \cup_{n \geq 0}\mathcal{L}^n,$$

kus

- $\mathcal{L}^0 = \{\epsilon\}$;
- $\mathcal{L}^n = \mathcal{L}^{n-1}\mathcal{L}$, *kui* $n > 0$

Definitsioon 26. *Olgu Σ_1 ja Σ_2 tähestikud. Kujutust $h : \Sigma_1^* \longrightarrow \Sigma_2^*$ nimetatakse* homomorfismiks, *kui*

- $h(\epsilon) = \epsilon$ *ja*
- $h(ax) = h(a)h(x)$ *iga* $x \in \Sigma_1^*$ *ja* $a \in \Sigma_1$ *korral.*

3.4. Grammatikad

Transleerimisprotsessi automatiseerimiseks vajatakse formaalset aparatuuri keele ja tema lausete fraasistruktuuri esitamiseks.

Matemaatilises mõttes on keel teatud tähestiku $T = \{a_0, a_1, \ldots, a_n\}$ stringide alamhulk

$$L = \{x \mid x \in T^*,\ P(x)\} \subseteq T^*.$$

Keele esitamine määrava predikaadi abil ei kirjelda süntaksianalüüsi ja teksti genereerimise protsesse, seepärast vaatleme edasises keelte defineerimise formalisme, mis esitavad keele tema lausete konstrueerimisalgoritmi kaudu.

Lisaks keele tähesikule Σ, mida edaspidi nimetame ka *terminaalide* tähestikuks, toome sisse metatähised fraaside tähistamiseks. Fraase tähistavate metasümbolite hulka nimetatakse *mitteterminaalide* tähestikuks. Kui pole öeldud teisiti, siis kasutame terminaalide tähistamiseks ladina tähestiku väiketähti ning numbrimärke ja mitteterminaalide tähistamiseks ladina tähestiku suurtähti. Kogu järgnevas käsitluses eeldatakse, et $\Sigma \cap N = \emptyset$.

Stringe tähestikus $V = \Sigma \cup N$ nimetame lausevormideks. Näites 12 esitatud lause lausevormideks on näiteks:

NF põhjustab NF halvenemise
N OMF temperatuur põhjustab NF
Sulami OMF N TF
:
:

Teksti genereerimisel kasutatakse lausevormi järkjärgulist teisendamist, kuni saadakse string, mis ei sisalda enam mitteterminaale. Teisendusreeglid esitatakse tavaliselt paaridena

$$\alpha \longrightarrow \beta,$$

kus α ja β tähistavad lausevorme, s.t. $\alpha, \beta \in V^*$.

Avaldist $\alpha \longrightarrow \beta$ nimetatakse produktsiooniks. Produktsiooni $\alpha \longrightarrow \beta$ olemasolu korral võib igas lausevormis, mis sisaldab alamsõna α, asendada see sõnaga β. Näiteks, kui on antud produktsioon

$$\text{NF põhjustab} \longrightarrow \text{NF põhjustab NF},$$

võib lausevormi

$$\text{NF põhjustab halvenemise}$$

asendada lausevormiga

$$\text{NF põhjustab NF halvenemise}.$$

Definitsioon 27. *Generatiivseks grammatikaks e. lihtsalt grammatikaks nimetatakse nelikut $G = (\Sigma, N, P, S_0)$, kus*

- *Σ on terminaalide tähestik;*
- *N on mitteterminaalide tähestik, kusjuures $\Sigma \cap N = \emptyset$;*
- *$P \subseteq V^*NV^* \times V^*$ on produktsioonide hulk, kus $V = \Sigma \cup N$;*
- *$S_0 \subseteq N$ on grammatika lähtesümbol.*

Näide 13. Grammatika $G_1 = (\{A, S\},\ \{0, 1\}\ P,\ S)$, kus P sisaldab järgmised produktsioonid:

$$P = \left\{ \begin{array}{l} S \longrightarrow 0A1 \\ 0A \longrightarrow 00A1 \\ A \longrightarrow \epsilon \end{array} \right\}$$

□

Definitsioon 28. *Lausevorm ψ on grammatikas G vahetult tuletatav lausevormist φ, kui $\varphi = \gamma\alpha\delta$ ja $\psi = \gamma\beta\delta$ $(\gamma, \beta \in V^*)$ ning grammatika G produktsioonide hulk P sisaldab produktsiooni $\alpha \longrightarrow \beta$.*

Vahetu tuletatavus on binaarne relatsioon stringide hulgal V^*, mida tähistatakse sümboliga $\Longrightarrow_G$. Indeksi G võib ära jätta, kui kontekstist on selge, millise grammatika suhtes tuletatavust vaadeldakse.

Näide 14. Näites 13 esitatud grammatikas kehtib seos $S \Longrightarrow^3 0011$. Tõepoolest $S \Longrightarrow 0A1 \Longrightarrow 00A11 \Longrightarrow 0011$. Kasutades matemaatilist induktsiooni saab näidata, et grammatikas G saab lähtesümbolist S tuletada n sammu jooksul sõna $0^{n-1}1^{n-1}$. □

Definitsioon 29. *Grammatika G poolt genereeritav keel on hulk*

$$L(G) = \{w \mid w \in \Sigma^*,\ s \Longrightarrow^* w\}.$$

Esituse lühiduse huvides kasutatakse allpool järgmisi tähistusi:

- $a, b, c, d, \ldots$ ja numbrimärgid $0, 1, \ldots, 9$ -terminaalid;
- $A, B, C, D, \ldots, S$ - mitteterminaalid, kusjuures S on algsümbol;
- $U, V, \ldots, Z$ - terminaalid või mitteterminaalid;
- $\alpha, \beta, \ldots,$ - stringid, mis võivad sisaldada nii terminaale kui ka mitteterminaale;
- $u, v, \ldots, z$ - ainult terminaalidest koosnevad stringid.

Toodud kokkulepped lubavad sageli grammatika esitamisel piirduda vaid produktsioonide hulga määratlemisega. Näiteks keele $L = \{a^n b^n c^n \mid n > 0\}$ genereeriks grammatika G, mille produktsioonide hulk

$$P_2 = \left\{ \begin{array}{ll} S \longrightarrow aSBC & bB \longrightarrow bb \\ S \longrightarrow aBC & bC \longrightarrow bc \\ CB \longrightarrow BC & cC \longrightarrow cc \\ aB \longrightarrow ab & \end{array} \right\}$$

Näide 15. Grammatika G poolt genereeritava keele omaduste tõestamine.

Olgu grammatika G määratud produktsioonidega

$$\begin{array}{ll} S \longrightarrow CD & Ab \longrightarrow bA \\ C \longrightarrow aCA & Ba \longrightarrow aB \\ C \longrightarrow bCB & Bb \longrightarrow bB \\ AD \longrightarrow aD & BD \longrightarrow bD \\ Aa \longrightarrow aA & C \longrightarrow \epsilon \\ & D \longrightarrow \epsilon. \end{array}$$

Väide 1. $L(G) = \{ww \mid w \in \{a, b\}^*\}$.

Väite tõestamiseks näitame kahe kuuluvusrelatsiooni kehtivust:

(a) $\{ww \mid w \in \{a, b\}^*\} \subseteq L(G)$

ja

(b) $L(G) \subseteq \{ww \mid w \in \{a, b\}^*\}$.

Väidet (a) on lihtne tõestada induktsiooni abil. Kõigepealt tuleb näidata, et

1. $S \Longrightarrow CD$
2. $n \geq 0$ korral
$$\begin{array}{rl} C & \Longrightarrow^n c_1 c_2 \ldots c_n C X_n X_{n-1} \ldots X_1 \\ & \Longrightarrow \; c_1 c_2 \ldots c_n X_n X_{n-1} \ldots X_1, \end{array}$$
kus $1 \leq i \leq n$ korral $c_i = a$ parajasti siis, kui $X_i = A$ ja $c_i = b$ parajasti siis, kui $X_i = B$.
3. $$\begin{array}{rll} X_n \ldots X_2 X_1 D & \Longrightarrow & X_n \ldots X_2 c_1 D \\ & \Longrightarrow^{n-1} & c_1 X_n \ldots X_2 D \\ & \Longrightarrow & c_1 X_n \ldots X_3 c_2 D \\ & \Longrightarrow^{n-2} & c_1 c_2 X_n \ldots X_3 \\ & \vdots & \\ & \Longrightarrow & c_1 c_2 \ldots c_{n-1} X_n D \\ & \Longrightarrow & c_1 c_2 \ldots c_{n-2} c_n D \\ & \Longrightarrow & c_1 c_2 \ldots c_n \end{array}$$

Kombineerides osaväiteid 1. - 3. saab näidata, et

$$S \Longrightarrow^* c_1 c_x \ldots c_n c_1 c_2 \ldots c_n.$$

Seega kehtib väide (a).

Väite (b) tõestus.
Defineerime kaks homomorfismi g ja h, nii et

$$g(a) = a, \;\; g(b) = b \;\; ja \;\; g(A) = g(B) = \epsilon$$

$$h(a) = h(b) = \epsilon, \;\; h(A) = a \;\; ja \;\; h(B) = b$$

Induktsiooni abil saab tõestada, et iga $m \geq 1$ korral

$$S \Longrightarrow^m \alpha = c_1 c_2 \ldots c_n U \beta V,$$

kus

- $c_i \in \{a, b\}$ iga $i = 1, 2, \ldots, n$ korral;
- U on kas C või ϵ;
- β on n sümboli pikkune string tähestikust $\{a, b, A, B\}$, nii et

  $$g(\beta) = c_1 \ldots c_i \quad h(b) = X_n X_{n-1} \ldots X_{i+1},$$

  kusjuures $X_j = A$, kui $c_j = a$ ja $X_j = B$, kui $c_j = b$, nii et $i < j \leq n$;
- V on kas D või ϵ.

Kuna keelde $L(G)$ kuuluvad vaid terminaalsed sõnad, siis

$$L(G) \subseteq \{ww \mid w \in \{a, b\}^*\}.$$

□

Ülesanne 1. *Millise keele genereerivad järgmised grammatikad:*

- $S \longrightarrow 0$
  $S \longrightarrow 1$
  $S \longrightarrow S0$

- $S \longrightarrow ()$
  $S \longrightarrow (S)$
  $S \longrightarrow SS$

- $S \longrightarrow ()$
  $S \longrightarrow (S)$
  $S \longrightarrow SSS$

- $S \longrightarrow 0B$
  $S \longrightarrow 1A$
  $A \longrightarrow 0$
  $A \longrightarrow 0S$
  $A \longrightarrow 1AA$
  $B \longrightarrow 1$
  $B \longrightarrow 1S$
  $B \longrightarrow 0BB$

Definitsioon 30. *Grammatikat $G = (S, N, P, S)$ nimetatakse*

- *0-tüüpi grammatikaks, kui talle pole seatud mingeid lisakitsendusi (vastava grammatikate klassi tähis on $\mathcal{L}_0$);*

– *1.-tüüpi grammatikaks e.* kontekstist sõltuvaks *grammatikaks, kui iga produktsiooni* $\alpha \longrightarrow \beta \in P$ *korral kehtib seos*

$$0 < \mid \alpha \mid \ \leq \ \mid \beta \mid, \tag{12}$$

välja arvatud juhul, kui $\epsilon \in L(G)$*, siis sisaldab produktsioonide hulk ühe elemendi, mis ei rahulda seost* (12)*:*

$$S \longrightarrow \epsilon$$

(grammatikate klassi tähis on $\mathcal{L}_1$*);*

– *2.-tüüpi grammatikaks e.* kontekstivabaks grammatikaks *(lühendatult KV-grammatikateks), kui iga produktsioon on kujul*

$$A \longrightarrow w,$$

kus

$$A \in N \text{ ja } w \in V^*$$

(grammatikate klassi tähis on $\mathcal{L}_2$*);*

– *3.-tüüpi grammatikaks e.* paremlineaarseks *(vastavalt* vasaklineaarseks*) grammatikaks, kui kõik produktsioonid on kujul*

$$A \longrightarrow bC \text{ või } A \longrightarrow b$$

(vastavalt $A \longrightarrow Cb$ *või* $A \longrightarrow b$*), kus* $A, C \in N$*,* $b \in \Sigma$ *või* $b = \epsilon$*.*
(Lineaarsete grammatikate klassi tähis on $\mathcal{L}_3$*).*

Omadus 1. $\mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \mathcal{L}_2 \supseteq \mathcal{L}_3$
(Otsene järeldus definitsioonist 30).

3.5. Regulaarsed avaldised

3.5.1. Regulaarsed hulgad

Definitsioon 31. *Olgu* Σ *lõplik tähestik.* Regulaarseks hulgaks *tähestikus* Σ *nimetatakse*

1. $\emptyset$ *(tühja hulka);*
2. $\{\epsilon\}$ *(hulka, mis koosneb vaid tühjast sõnast);*
3. $\{a\}$*, kus* $a \in \Sigma$*;*
4. *hulki* $P \cup Q, PQ$ *ja* Q^**, kui* P *ja* Q *on regulaarsed hulgad;*

ühtegi hulka, mis pole konstrueeritav reeglite 1. - 4. põhjal, ei nimetata regulaarseks.

Regulaarsete hulkade mugavamaks tähistamiseks kasutatakse nn. *regulaarseid avaldisi.*

Definitsioon 32. *Regulaarne avaldis on määratletud järgmiste rekursiivsete reeglitega 1. - 5.:*

1. $\emptyset$ *on regulaarne avaldis, mis tähistab tühja hulka;*
2. e *on regulaarne avaldis, mis tähistab hulka* $\{\epsilon\}$*;*
3. a *on regulaarne avaldis, mis tähistab hulka* $\{a\}$*, kus* $a \in \Sigma$*;*
4. *kui* p *ja* q *on regulaarsed avaldised, mis tähistavad vastavalt regulaarseid hulki* P *ja* Q*, siis on regulaarsed avaldised ka*
   – $(p + q)$ *- tähistab hulka* $P \cup Q$*;*
   – (pq) *- tähistab hulka* PQ*;*
   – $(p)^*$ *- tähistab hulka* P^**;*

5. kõik muud avaldised ei ole regulaarsed.

Järgnevas eeldatakse et regulaarse avaldise tehetest kõrgeima prioriteediga on operatsioon *, seejärel tuleb konkatenatsioon ning lõpuks operatsioon $+$. Sulge tarvitame järgnevas vaid seal, kus tehete järjekord pole kooskõlas loetletud prioriteetidega. Avaldis p^+ on lühend avaldise pp^* esitamiseks.

Regulaarseid avaldisi loetakse võrdseiks ($=$), kui nad tähistavad üht ja sama hulka. Iga regulaarse hulga jaoks leidub teda esitav regulaarne avaldis.

Näide 16. Avaldis 01 esitab üheelemendilise hulga $\{01\}$.
Avaldis $(a+b)(a+b+0+1)^*$ esitab keele

$$L = \{cw \mid w \in \{a,b,0,1\}^* \text{ ja } (c = a \text{ või } c = b)\}.$$

Avaldis $(0+1)^*$ esitab keele

$$L = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}.$$

Avaldis $(0+1)^*001$ esitab keele

$$L = \{001, 0001, 1001, 00001, 01001, \ldots\}.$$

□

Lemma 1. *Regulaarsed hulgad $\emptyset$, $\{\epsilon\}$ ja $\{a\}$, kus $a \in \Sigma$, on paremlineaarsed keeled.*

Tõestus.

- $G = (\Sigma, \{S\}, \emptyset, S)$ on paremlineaarne grammatika tühja keele $\mathcal{L}(G) = \emptyset$ genereerimiseks;
- $G = (\Sigma, \{S\}, \{S \longrightarrow \epsilon\}, S)$ on paremlineaarne grammatika, mille korral
$$\mathcal{L}(G) = \{\epsilon\}$$
- $G_a = (\Sigma, \{S\}, \{S \longrightarrow a\}, S)$, kus $a \in \Sigma$ on grammatika keele $\mathcal{L}(G_a) = \{a\}$ genereerimiseks.

□

Lemma 2. *Kui L_1 ja L_2 on paremlineaarsed keeled tähestikus Σ, siis on paremlineaarsed keeled ka*

- $L_1 \cup L_2$;
- $L_1 L_2$ *ja*
- L_1^*.

Tõestus. Vastavalt eeldustele leiduvad paremlineaarsed grammatikad
$G_1 = (\Sigma, N_1, P_1, S_1)$ ja $G_2 = (\Sigma, N_2, P_2, S_2)$, nii et $L_1 = \mathcal{L}(G_1)$ ja $L_2 = \mathcal{L}(G_2)$. Eeldame, et $N_1 \cap N_2 = \emptyset$. Konstrueerime grammatikad G_3, G_4 ja G_5 vastavalt keelte $L_1 \cup L_2$, $L_1 L_2$ ja L_1^* genereerimiseks.

1. Olgu $G_3 = (\Sigma, N_1 \cup N_2 \cup \{S_3\},\ P_1 \cup P_2 \cup \{S_3 \longrightarrow S_1, S_3 \longrightarrow S_2\}, S_3)$,
kusjuures S_3 on uus mitteterminal $S_3 \notin N_1 \cup N_2$.

   Iga sõna $w \in L_1$ korral leidub tuletus
$$S_3 \Longrightarrow_{G_3} S_1 \Longrightarrow^+_{G_3} w,$$
$$\text{kuna } S_1 \Longrightarrow^+_{G_1} w.$$
   Iga sõna $w \in L_2$ korral leidub tuletus
$$S_3 \Longrightarrow_{G_3} S_2 \Longrightarrow^+_{G_3} w,$$
$$\text{kuna } S_2 \Longrightarrow^+_{G_2} w.$$
   Toodud tuletustest järeldubki, et paremlineaarne grammatika G genereerib keele $\mathcal{L}(G_3) = L_1 \cup L_2$.

2. Olgu grammatika $G_4 = (\Sigma, N_1 \cup N_2, P_4, S_1)$, kus produktsioonide hulk

$$P_4 = P_2 \cup \{A \longrightarrow aB \mid A \longrightarrow aB \in P_1\} \cup \{A \longrightarrow aS_2 \mid A \longrightarrow a \in P_1\}.$$

Kui $S_1 \Longrightarrow^+_{G_1} w$, siis $S_1 \Longrightarrow^+_{G_4} wS_2$.
Kui $S_2 \Longrightarrow^+_{G_2} x$, siis $S_2 \Longrightarrow^+_{G_4} x$.
Järelikult $S_1 \Longrightarrow^+_{G_4} wx$ ja $\mathcal{L}(G_1)\mathcal{L}(G_2) \subseteq L(G_4)$.
Vaatleme nüüd tuletust $S_1 \Longrightarrow^+_{G_2} w$. Kuna P_4 ei sisalda ühtegi reeglit $A \longrightarrow a \in P_1$, siis on viimane tuletus esitatav kujul

$$S_1 \Longrightarrow^+_{G_4} xS_2 \Longrightarrow^+_{G_4} xy = \omega.$$

Vastavalt grammatika G_4 konstrueerimise reeglitele on alamsõna x genereerimiseks kasutatud hulga P_1 produktsioone ja alamsõna y genereerimiseks hulga P_2 produktsioone.
Järelikult $L(G_4) \subseteq \mathcal{L}(G_1)\mathcal{L}(G_2)$ ning $\mathcal{L}(G_4) = L_1L_2$.

3. Olgu grammatika $G_5 = (\Sigma, N_1 \cup \{S_5\}, P_5, S_5)$ selline, et $S_5 \notin N_1$ ja produktsioonide hulk

$$P_5 = P_1 \cup \{A \longrightarrow aS_5 \mid A \longrightarrow a \in P_1\} \cup \{S_5 \longrightarrow S_1, S_5 \longrightarrow \epsilon\}.$$

Analoogiliselt juhtumiga 2 saab näidata, et

$$S_5 \Longrightarrow^+_{G_5} x_1S_5 \Longrightarrow^+_{G_5} x_1x_2S_5 \Longrightarrow^+_{G_5} \ldots \Longrightarrow^+_{G_5} x_1x_2\ldots x_{n-1}S_5 \Longrightarrow^+ x_1x_2\ldots x_n$$

siis ja ainult siis, kui $S_1 \Longrightarrow^+_{G_1} x_1, \ldots, S_1 \Longrightarrow^+_{G_1} x_n$.
Järelikult $\mathcal{L}(G_5) = L_1^*$.

□

Lemma 1 ja lemma 2 otseseks järelduseks on järgmine teoreem.

Teoreem 3. *Regulaarne hulk on genereeritav paremlineaarse grammatikaga.*

3.5.2. Lõplikud automaadid

Grammatika või regulaarse avaldisega esitatud keele lausete genereerimise ülesandega duaalne ülesanne on lausete *aktsepteerimise ülesanne*:

> "Antud sõna x ja grammatika G (või regulaarse avaldise α) põhjal teha kindlaks, kas x kuulub grammatika G poolt genereeritavasse (avaldisega α määratud) keelde".

Regulaarsete hulkade aktsepteerimise ülesande lahendamiseks võib kasutada lõplikku automaati.

Definitsioon 33. *Lõplik automaat on viisik $M = (\Sigma, Q, \delta, Q_0, F)$, kus*

1. *Σ - sisendtähestik;*
2. *Q - olekusümbolite lõplik tähestik;*
3. *$\delta : \Sigma \times Q \longrightarrow P(Q)$ - üleminekufunktsioon;*
4. *$Q_0 \subseteq Q$ - lähteolekute hulk;*
5. *$F \subseteq Q$ - lõppolekute (aktsepteerivate olekute) hulk.*

Kui $\mid \delta(a,q) \mid \; = 1$ iga $a \in \Sigma$ ja $q \in Q$ korral ja $Q_0 = \{q_0\}$, nimetatakse lõplikku automaati M deterministlikuks, vastasel korral mittedeterministlikuks.

Lõpliku automaadi $M = (\Sigma, Q, \delta, Q_0, F)$ konfiguratsiooniks nimetatakse paari

$$(w, q) \in \Sigma^* \times Q.$$

Konfiguratsiooni kujul (w, q_0), kus $q_0 \in Q_0$ nimetatakse lõpliku automaadi M lähtekonfiguratsiooniks ning konfiguratsiooni (ϵ, r), kus $r \in F$ lõppkonfiguratsiooniks.

Definitsioon 34. *Lõpliku automaadi töötaktiks nimetatakse binaarset seost* $\vdash$ *konfiguratsioonide hulgal:*

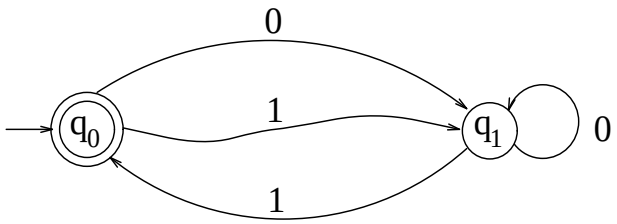
$$(aw, q_1) \vdash (w, q_2)$$

kehtib parajasti siis, kui $q_2 \in \delta(a, q_1)$.

Näide 17. Olgu $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1\}$ ning $Q_0 = F = \{q_0\}$. Regulaarse hulga $((0+1)0^*1)^*$ aktsepteerib lõplik (deterministlik) automaat, mille üleminekufunktsioon on esitatud järgmise tabelina:

Q \ Σ	0	1
→ q_0	q_1	q_1
q_1	q_1	q_0

Lõplike automaatide esitamiseks kasutatakse ka graafilist meetodit:



Antud automaat on deterministlik.

□

Definitsioon 35. *Lõpliku automaadi* $M = (\Sigma, Q, \delta, Q_0, F)$ *poolt aktsepteeritsv keel*

$$T(M) = \{w \mid (w, q_0) \vdash^* (\epsilon, r),\ q_0 \in Q_0,\ r \in F\}.$$

Teoreem 4. *Iga lõpliku automaadi poolt aktsepteeritav keel on paremlineaarne.*

Tõestus. Olgu $T(M)$ lõpliku automaadi $M = (\Sigma, Q, \delta, Q_0, F)$ poolt aktsepteeritav keel. Konstrueerime paremlineaarse grammatika $G = (\Sigma, Q, P, S)$, kus produktsioonide hulk

$$P = \{q \longrightarrow aq' \mid q' \in \delta(a, q)\} \cup \{q \longrightarrow a \mid \delta(a, q) \in F\} \cup \{S \longrightarrow q_0 \mid q_0 \in Q_0\}$$

Kasutades induktsiooni on lihtne näidata, et

$$q \Longrightarrow_G^* wq'$$

parajasti siis, kui

$$(w, q) \vdash^* (\epsilon, q').$$

Sellest järeldub, et $S \Longrightarrow^* w$ parajasti siis, kui $w \in T(M)$. Kuna G on paremlineaarne grammatika, siis $T(M)$ on paremlineaarne keel.

□

Teoreem 5. *Iga lõpliku automaadi poolt aktsepteeritav keel on regulaarne hulk.*

Tõestus. Olgu $L = T(M)$ lõpliku automaadi $M = (\Sigma, q, \delta, Q_0, F)$ korral, kus $Q = \{q_0, q_1, \ldots q_n\}$.

Defineerime kõigi stringide hulga, mis viivad automaadi M olekust q_i olekusse q_j:

$$R_{ij} = \{w \mid (w, q_i) \vdash^* (\epsilon, q_j)\}.$$

Kuna $L = \cup\{R_{ij} \mid q_i \in Q_0,\ q_j \in F\}$, siis piisab tõestuseks näidata, et iga $i, j \leq n$ korral on hulk R_{ij} regulaarne.

Hulga R_{ij} regulaarsuse näitamiseks toome sisse abihulgad R^k_{ij}, mis koosnevad neist stringidest, mille korral automaat M läheb olekust q_i olekusse q_j, ilma et vahepealsete taktide jooksul läbitaks olekuid $q_k, q_{k+1}, \ldots, q_n$.

Märkigem, et $R^0_{ij} = \{a \in \Sigma \mid q_j \in \delta(q_i, a)\}$ on lõplikud ja seega regulaarsed.

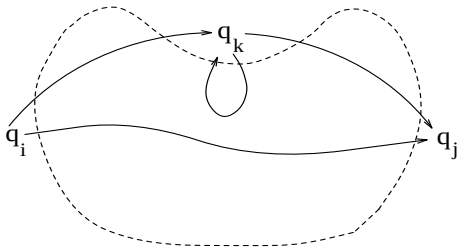
Teiselt poolt aga $R^{n+1}_{ij} = R_{ij}$.

Eeldame, et R^k_{ij} on regulaarne mingi k ja iga i, j korral.

Näitame, et R^{k+1}_{ij} on regulaarne suvalise i ja j jaoks.

Olgu sõna $w \in R^{k+1}_{ij}$ (vt. joonis). Siis on võimalikud kaks juhtu:

1. sõna w analüüsile vastav tee automaadi M graafis ei läbi olekut q_k;
2. olek q_k kuulub sõnale w vastavale teele.



olekud q_0 kuni q_{k-1}

Esimesel juhul $w \in R^k_{ij}$, mis on eelduse põhjal regulaarne.

Teisel juhul $w = w_1 w_2 \ldots w_m$, $m \geq 2$, nii et

w_1 vastab teele q_1-st q_k läbi olekute hulgast $\{q_1, \ldots, q_{k-1}\}$;
w_m vastab teele q_k-st q_j läbi olekute hulgast $\{q_1, \ldots, q_{k-1}\}$ ja
$w_n (1 < n < m)$ vastab teele q_k-st q_k-sse läbi olekute $\{q_1, \ldots, q_{k-1}\}$.

Seega teisel juhul

$$w \in R^k_{ik}\ (R^k_{kk})^*\ R^k_{kj}.$$

Järelikult $R^{k+1}_{ij} = R^k_{ij} \cup R^k_{ik}\ (R^k_{kk})^*\ R^k_{kj}$ ning induktsiooni eeldusest järeldub, et ka R^{k+1}_{ij} on iga i ja j korral regulaarne.

□

Teoreem 6. *Iga paremlineaarne keel on aktsepteeritav (mittedeterministliku) lõpliku automaadi abil.*

Tõestus. Olgu keel L genereeritav paremlineaarse grammatikaga $G = (\Sigma, N, P, S)$. Moodustame lõpliku automaadi

$$M = (\Sigma, N \cup \{F\}, \delta, \{S\}, \{F\}),$$

kus

- $F \notin N$;
- $B \in \delta(a, A)$ parajasti siis, kui $A \longrightarrow aB \in P$;
- $F \in \delta(a, A)$ parajasti siis, kui $A \longrightarrow a \in P$.

Induktsiooni abil on lihtne näidata, et

$$S \Longrightarrow_G^* wA \Longrightarrow_G waB \Longrightarrow_G^* waw'C \Longrightarrow_G waw'b$$

parajasti siis, kui automaadi M jaoks

$$(waw'bS) \vdash^* (aw'b, A) \vdash (w'b, B) \vdash^* (b, C) \vdash (\epsilon, F).$$

Seega $L = T(M)$.

□

Teoreem 7. *Iga mittedeterminstliku lõpliku automaadi N jaoks leidub selline deterministlik lõplik automaat M, nii et $T(M) = T(N)$.*

Tõestus. Olgu $N = (\Sigma, Q, \delta, Q_0, F)$. Moodustame deterministliku automaadi

$$M = (\Sigma, Q', \delta', Q_0', F'), \text{kus}$$

1. $Q' = \mathcal{P}(Q)$;
2. $\delta'(a, p) = \cup_{q \in P} \{\delta(a, q)\}$;
3. $Q_0' = \{q_0 \mid q_0 \in Q_0\}$;
4. $F' = \{q \mid q \in Q, q \cap F \neq \emptyset\}$.

Toodud konstruktsiooni tingimusest 2 järeldub, et automaadi N korral kehtib

$$(aw, p) \vdash (w, q^*) \text{ e. } q \in \delta(a, p)$$

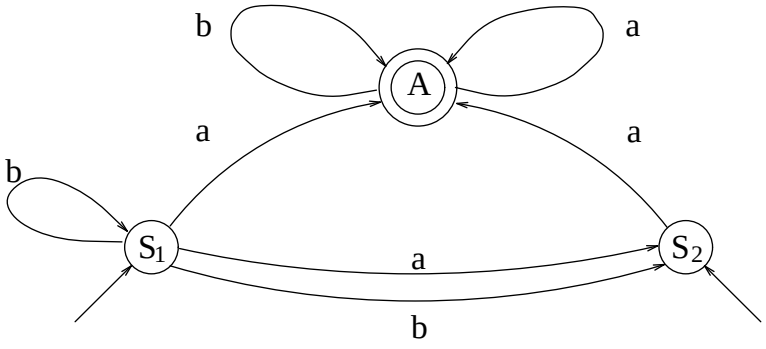
parajasti siis, kui automaadi M korral

$$(aw, P) \vdash (w, Q)$$

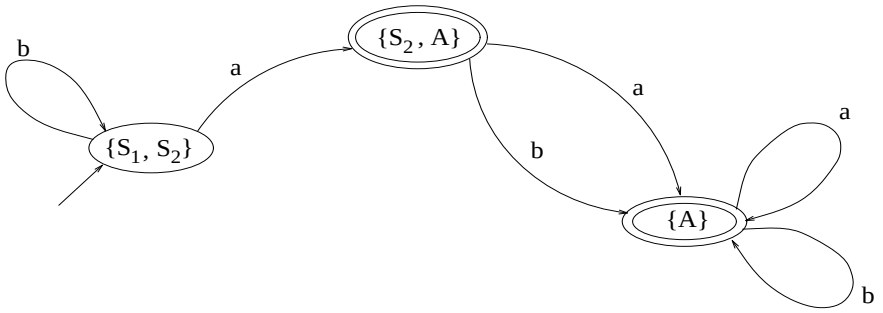
ning $p \in P$ ja $q \in Q$.

Induktsiooni abil saab lihtsalt näidata, et $T(N) = T(M)$.

Näide 18. Olgu antud mittedeterministlik lõplik automaat:



Kasutades teoreemi 7 tõestuses esitatud konstruktsiooni ning elimineerides seejärel liigsed olekud, saame eeltooduga ekvivalentse deterministliku lõpliku automaadi.



□

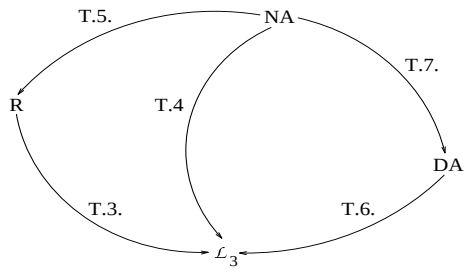
Teoreem 8. *Järgmised kolm väidet on ekvivalentsed:*

1. *L on regulaarne hulk;*
2. *L on paremlineaarne keel;*
3. *L on aktsepteeritav deterministliku lõpliku automaadi abil.*

Tõestus. Otsene järeldus teoreemidest 3 - 7. Kui kasutada järgmisi tähistusi

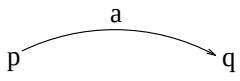
R - regulaarsed hulgad
$\mathcal{L}_3$ - paremlineaarsed keeled
NA - mittedeterministlike automaatidega aktsepteeritavad keeled
DA - deterministlike automaatidega aktsepteeritavad keeled,

võib eelmised teoreemid graafiliselt esitada järgmise diagrammi abil.

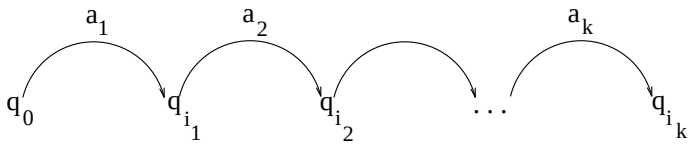


Teoreem 9. (Tarvilikkuse tingimus keele regulaarsuseks). *Olgu deterministlik lõplik automaat, millel on n olekut. Iga sõna $z \in T(M)$, mille korral $|\ z\ | \geq n$ on esitatav kujul $z = uvw$, nii et iga $j \geq 0$ korral $uv^j w \in T(M)$.*

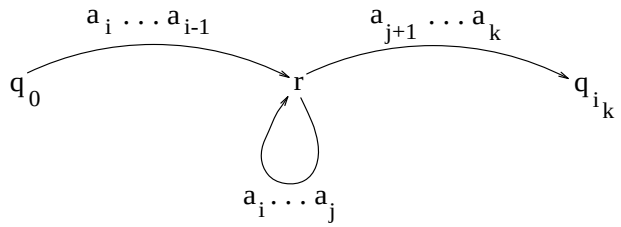
Tõestus. Automaadi M käitumist sümboli a skaneerimisel võib graafiliselt esitada kujul



kui $\delta(a, p) = q$.
Sõna $z = a_1, a_2, \ldots, a_k$ aktsepteerimine on esitatav graafina



Kui $|z| \geq n$, peab toodud jadas mõni olek esinema korduvalt. Olgu selliseks olekuks r. Järelikult saab automaadi esitada kujul



Valime nüüd $u = a_1 \ldots a_{i-1}$, $v = a_i \ldots a_j$ ja $w = a_{j+1} \ldots a_k$. Lihtne on näha, et keelde $T(M)$ kuuluvad sõnad uw, uvw, $uvvw$, $uvvvw$, ...

Teoreem 9 võimaldab sageli lihtsalt näidata, et teatud keel ei ole regulaarne (paremlineaarne).

Näide 19. Keel $L = \{0^n 1^n \mid n > 1\}$ ei ole paremlineaarne. Oletame, et see keel on regulaarne. Siis küllalt suure n korral $0^n 1^n = uvw$ nii, et sõnad $uv^i w$, $i = 0, 1, 2, \ldots$ oleksid kujul $0^m 1^m$. Kui v sisaldab vaid üht tähte ($v = 0^k$ või $v = 1^k$), siis sõnas $uv^0 w = uw$ ei ole nullide ja ühtede arv enam võrdne, kui aga v sisaldab mõlemaid, ei ole sõnas $uvvw$ kõik nullid enne ühtesid.

Järelikult $L \notin \mathcal{L}_3$.

□

Järeldus 1. *Sisalduvus $\mathcal{L}_3 \subset \mathcal{L}_2$ on range.*

3.6. Kontekstivabad grammatikad

Lihtsa KV-grammatika näitena vaatleme kahte produktsiooni sisaldavat grammatikat.

$$S \longrightarrow 0S1$$

$$S \longrightarrow 01.$$

See grammatika genereerib keele $L = \{0^n 1^n \mid n > 0\}$. Järelikult $L \in \mathcal{L}_2$. Näitest 19 selgub aga, et $L \notin \mathcal{L}_3$.

KV-keelte omaduste analüüsimisel kasutatakse sageli süntaksipuid.

3.6.1. Süntaksipuud

Iga järjestatud puu $T = (A, R)$, mille tippude märgistus on antud kujutusega f, on esitatav termina vastavalt reeglitele:

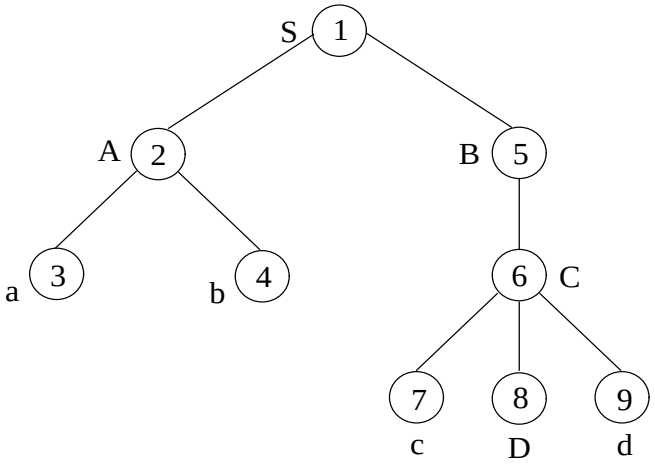
– kui $a \in A$ on puu T terminaalne tipp, siis märgend $M = f(a)$ on term;

– kui $a \in A$ on puu T mitteterminaalne tipp märgendiga $M = f(a)$, mille vahetuid alampuid, loetletuna vasakult paremale, tähistavad termid $t_1, \ldots, t_n$, siis on avaldis $M(t_1, \ldots, t_n)$ term, mis tähistab puu T alampuud juurega a.

Toodud term on saadav puu eesjärjekorrast lrep(T), asendades selles tipud neile vastavate märgenditega.

Puu T *krooniks* $Kr(T)$ nimetatakse stringi, mis saadakse tema terminaalsete tippude märgenditest, kirjutades need järjestikku alates kõige vasakpoolsemast ning lõpetades parempoolseimaga. Lähtudes puud esitavast termist saab tema krooni, kui kustutada termist kõik need märgendid, millele järgneb vahetult avanev sulg ning seejärel kõik sulud ja komad.

Näide 20. Puud t:



esitab term $t = S(A(a, b),\ B(C(c, D, d)))$, mille kroon $Kr(t) = abcDd$.

□

Situatsioonis, kus meid ei huvita puu t sisemised tipud, kasutame tema tähistusena termi $A[Kr(t)]$, kus A on puu t juure märgend. Eelmises näites toodud puu tähistame sel juhul termiga $S[abcDd]$.

Puus t_1 terminaalse tipu A asendamist puuga T_2, mille juur on märgendatud sümboliga A, tähistame $t = t_1\{A/t_2\}$. Lihtne on näha, et kui $t_1 = S[\alpha A\gamma]$ ja $t_2 = A[\beta]$, siis $t = S[\alpha\beta\gamma]$.

Puud, mis koosneb vaid juurest ning terminaalsetest tippudest, nimetame edaspidi *elementaarpuuks*.

Iga puu T on üheselt esitatav oma elementaarpuude nimistuga $E(T)$. Näites 20 on esitatud puu t elementaarpuude korteež

$$E(t) =< S(A, B),\ A(a, b),\ B(C),\ C(c, D, d) >$$

Korteežidevaheliste ning korteežide ja elementide vaheliste kuuluvusseoste esitamiseks kasutame hulgateooria tähiseid. Seega, kui α ja β on korteežid ning x element, kasutame tähistusi

$x \in \alpha -$ x on korteeži α elemendiks

$\alpha \subset \beta -$ α on β alamkorteež, e. seosest $x \in \alpha$ järeldub, et $x \in \beta$

Märgendatud järjestatud elementaarpuid võib kasutada KV-grammatika produktsioonide esitamiseks:

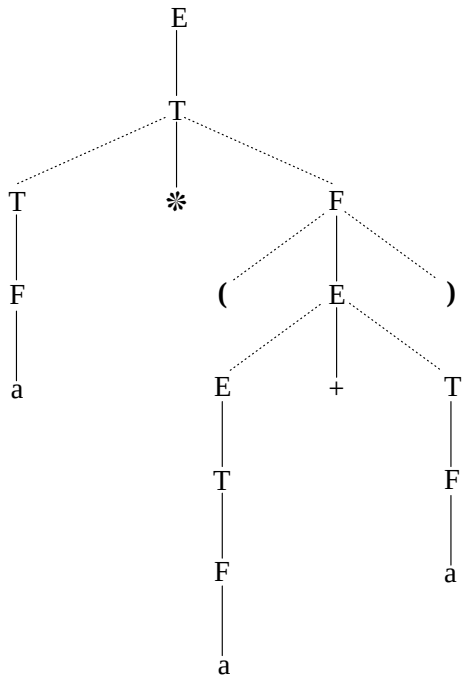
puu $A(x_1 \ldots x_n)$ kujutagu produktsiooni $A \longrightarrow x_1 \ldots x_n$. Asjaolu, et elementaarpuu r esitab produktsiooni p tähistame $r \mapsto p$.

Definitsioon 36. *Süntaksipuuks KV-grammatikas* $G = (\Sigma, N, P, S)$ *nimetatakse märgendatud järjestatud puud* t*, kui iga* $r \in E(t)$ *korral* $R \mapsto p, p \in P$.

Näide 21. KV-grammatikas $G = (\{a, +, *, (,)\}, \{E, T, F\}, P, E)$, kus

$$P = \left\{ \begin{array}{ll} E \longrightarrow E + T & T \longrightarrow F \\ E \longrightarrow T & F \longrightarrow (E) \\ T \longrightarrow T * F & F \longrightarrow a \end{array} \right\}$$

on süntaksipuu



Termina oleks see puu kujul

$$t = E(T(T(F(a)), *, F((, E(E(T(F(a))), +, T(F(a))),)))).$$

Kroon $Kr(t) = a * (a + a)$. See puu võib olla esitatud kujul

$$E[a * (a + a)]$$

□

Teoreem 10. *KV-grammatikas $G = (\Sigma, N, P, S)$ on mitteterminaalist $A \in N$ tuletatav lausevorm α parajasti siis, kui grammatikas G leidub süntaksipuu $A[\alpha]$.*

Tõestus.

Tarvilikkus. Kui string α on tuletatav 0 sammuga, siis $\alpha = A$ ($A \Longrightarrow^0 A$) ning sellisele tuletusele võib vastavusse seada ühest tipust (märgendiga A) koosneva puu $A[A]$. Kuna antud juhul $E(A[A]) = \emptyset$, on tegu süntaksipuuga.

Oletame, et igale tuletusele $B \Longrightarrow^k \beta$ seatakse vastavusse süntaksipuu $B[\beta]$. Iga $(k+1)$-sammulise tuletuse

$$A \Longrightarrow \gamma_1 B \gamma_2 \Longrightarrow^k \gamma_1 \beta \gamma_2 = \alpha$$

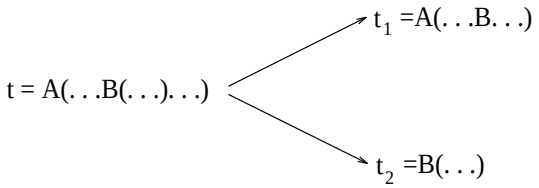
korral leidub grammatikas G produktsioon $A \longrightarrow \gamma_1 B \gamma_2$, mis on esitatav elementaarpuuna $A(\overline{\gamma}_1, B, \overline{\gamma}_2)$, kus $\overline{\gamma}_1$ tähistab stringi γ_1 koosseisu kuuluvate sümbolite loetelu. Järelikult saab moodustada süntaksipuu $A(\overline{\gamma}_1, B[\beta], \overline{\gamma}_2) = A[\alpha]$.

Piisavus. Olgu antud süntaksipuu $A[A]$, mis koosneb vaid ühest tipust A, siis talle vastav tuletus on $A \Longrightarrow^0 A$.

Oletame, et kõigi nende süntaksipuude $A[\alpha]$ jaoks, mille termis on k avavat sulgu, on olemas tuletus $A \Longrightarrow^* \alpha$.

Olgu süntaksipuu $A[\alpha]$ termis $k+1$ avanevat sulgu. Valime neist ühe (mitte esimese) ning eraldame termist valitud sulule vastava alamtermi. Väljaeraldatud alamtermi asemele kirjutame lähtetermi valitud sulule eelneva mitteterminaali.

Graafiliselt võib kirjeldatud teisendust esitada järgmiselt:



Termid t_1 ja t_2 esitavad süntaksipuid $A[\gamma_1 B \gamma_2]$ ning $B[\beta]$ grammatikas G, sest nad koosnevad endiselt produktsioonidele vastavatest elementaarpuudest. Paneme seejuures tähele, et $\alpha = \gamma_1\beta\gamma_2$. Teiselt poolt sisaldavad mõlemad termid vähem kui k avanevat sulgu ja seega leiduvad tuletused $A \Longrightarrow^* \gamma_1 B \gamma_2$ ($B \in Kr(t_1)$, sest ta asendab terviklikku alamtermi ning talle ei saa järgneda avanev sulg) ja $B \Longrightarrow^* \beta$. Järelikult saab grammatikas G esitada tuletuse

$$A \Longrightarrow^* \gamma_1 B \gamma_2 \Longrightarrow^* \gamma_1 \beta \gamma_2 = \alpha.$$

□

Definitsioon 37. *Täielikku süntaksipuud t (s.o. süntaksipuud, mida ei saa laiendada - puu juur on märgendatud grammatika lähtesümboliga ja kõik lehed on märgendatud terminaalidega), mille kroon* $\mathrm{Kr}(t) = x \in \Sigma^*$ *nimetatakse sõna x* tuletuspuuks.

Teoreemist 10 järeldub, et sõna x kuulub keelde $\mathcal{L}(G)$ parajasti siis, kui grammatikas G leidub tuletuspuu t, nii et $Kr(t) = x$.

Teoreemi 10 tarvilikkuse tõestuses esitatud konstruktsioon võimaldab igale tuletusele grammatikas G üheselt leida vastava tuletuspuu. Tuletuspuu näitab, millistes grammatilistes seostes on genereeritava lause liikmed ning on ühtlasi ka lause semantika e. tähenduse väljendajaks. Juhul kui lausele saab ehitada mitu tuletuspuud, on lausel ka reeglina mitu erinevat tähendust. Programmeerimiskeelte korral, kui nõutakse tekstide ühetähenduslikkust, peab igal lausel olema vaid üks tuletuspuu.

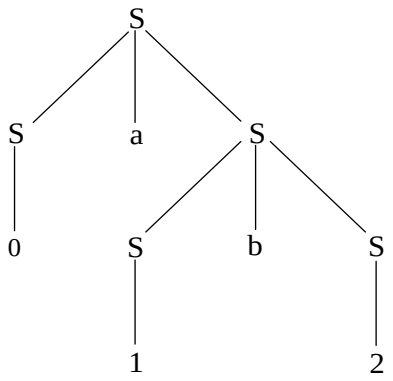
Teatud juhtudel on KV-keele lausetel mitu erinevat tuletust ning sellega kaasneb ka erinevate tuletuspuude olemasolu. Näiteks, kui grammatika on määratud produktsioonidega

$$\begin{aligned} S &\longrightarrow SaS \\ S &\longrightarrow SbS \\ S &\longrightarrow 0 \\ S &\longrightarrow 1 \\ S &\longrightarrow 2 \end{aligned}$$

saab sõna $0a1b2$ jaoks koostada *vasaktuletuse* (igal tuletuse sammul teisendatakse kõige vasakpoolsemat mitteterminaali):

$$S \Longrightarrow SaS \Longrightarrow 0aS \Longrightarrow 0aSbS \Longrightarrow 0a1bS \Longrightarrow 0a1b2.$$

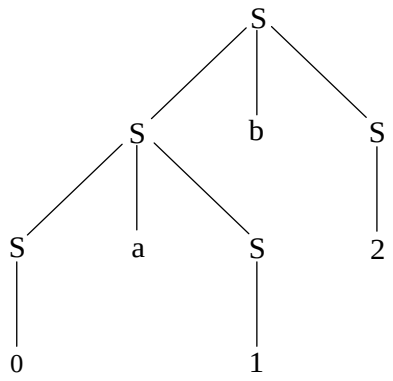
Vastav tuletuspuu omab kuju



Sama sõna *paremtuletus* (igal sammul asendatakse kõige parempoolsem mitteterminaal) saadakse tuletus järgmiselt:

$$S \Longrightarrow SbS \Longrightarrow Sb2 \Longrightarrow SaSb2 \Longrightarrow Sa1b \Longrightarrow 0a1b2.$$

Vastav tuletuspuu oleks erinev eelnevast:



Definitsioon 38. *KV-grammatikat G, mille korral leidub sõnal $x \in \mathcal{L}(G)$ mitu erinevat tuletuspuud, nimetatakse* mitmeseks.

Teatud mitmeste grammatikate korral leidub ekvivalentne ühene grammatika (isegi kui ühel sõnal on mitu erinevat tuletust, vastab neile kõigile üks ja seesama tuletuspuu).

Viimatiesitatud grammatika saab teisendada kujule

$$\begin{array}{ll} S \longrightarrow SaA & A \longrightarrow SbA \\ S \longrightarrow 0 & A \longrightarrow 2 \\ S \longrightarrow 1 & \end{array}$$

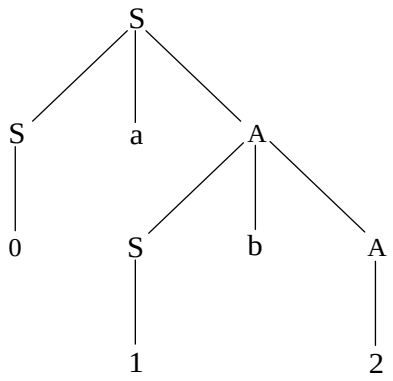
Sel juhul on nii vasaktuletuse

$$S \Longrightarrow SaA \Longrightarrow 0aA \Longrightarrow 0aSbA \Longrightarrow 0a1bA \Longrightarrow 0a1b2$$

kui ka paremtuletuse

$$S \Longrightarrow SaA \Longrightarrow SaSbA \Longrightarrow SaSb2 \Longrightarrow Sa1b2 \Longrightarrow 0a1b2$$

korral tegu ühe ja sama tuletuspuuga



Definitsioon 39. *KV-keelt L, millel leidub ühene genereeriv grammatika G ($L = \mathcal{L}(G)$) nimetatakse* üheseks, *vastasel korral* mitmeseks.

Viimatinimetatud keel on näiteks ühene. Mitmese kontekstivaba keele näitena võib tuua hulga

$$L = \{a^m b^n c^l \mid n = m \text{ või } m = 1\}.$$

Toodud näidetest võib jääda ettekujutus, et mitmese keele korral saab soovitud tuletuspuu, kui kasutada mngit kindlat tuletusmeetodit (näit. vasak- või paremtuletust). Nii see siiski pole.

Igal sõnal $x \in \mathcal{L}(G)$ leidub grammatikas G vasaktuletus (paremtuletus). Iga tuletuse saab asendada vasak- või paremtuletusega, muutes genereerimisel produktsioonide kasutamise järjekorda. Veendumaks, et produktsioonide kasutamisjärjekorra muutmine ei mõjuta tuletatava sõna kuju, võrrelge kahte järgmist tuletust:

$$S \Longrightarrow^* \alpha_1 A \beta_1 B \gamma_1 \Longrightarrow \alpha_1 A \beta_1 \delta_1 \gamma_1 \Longrightarrow^* \alpha_2 A \beta_2 \delta_2 \gamma_2 \Longrightarrow \alpha_2 \psi_1 \beta_2 \delta_2 \gamma_2 \Longrightarrow^* x$$

ja

$$S \Longrightarrow^* \alpha_1 A \beta_1 B \gamma_1 \Longrightarrow \alpha_1 \psi_1 \beta_1 B \gamma_1 \Longrightarrow^* \alpha_2 \psi_1 \beta_2 B \gamma_2 \Longrightarrow \alpha_2 \psi_1 \beta_2 \delta_2 \gamma_2 \Longrightarrow^* x.$$

Seega on grammatika ühesuse (mitmesuse) definitsioon (Def.39) ekvivalentne järgmise määratlusega.

Definitsioon 40. *KV-grammatika G on ühene, kui ei leidu sõna $x \in \mathcal{L}(G)$, nii et tema erinevad vasaktuletused (paremtuletused) omaksid erinevaid tuletuspuid.*

Näide 22. Mitmese KV-grammatika

$$\begin{aligned} S &\longrightarrow SbS \\ S &\longrightarrow SaS \\ S &\longrightarrow c \end{aligned}$$

korral leidub sõnal cacbc kaks erinevat vasaktuletust:

$$S \Longrightarrow SaS \Longrightarrow caS \Longrightarrow caSbS \longrightarrow cacbS \Longrightarrow cacbc$$

ja

$$S \Longrightarrow SbS \Longrightarrow SaSbS \Longrightarrow caSbS \Longrightarrow cacbS \Longrightarrow cacbc.$$

Neile tuletustele vastavad ka erinevad tuletuspuud.

□

3.6.2. KV-grammatikate teisendamine

Käesolevas alajaotuses vaatleme teisendusi, mis võimaldavad suvalist KV-grammatikat lihtsustada ning viia ta standardsele kujule.

Algoritm 1. (Kas keel $\mathcal{L}(G)$ on tühi?).
Sisend: KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: "JAH", kui $\mathcal{L}(G) \neq 0$, "EI" vastasel juhul.
Meetod: Konstrueerida rekursiivselt hulgad $N_0, N_1, \ldots$ vastavalt punktidele 1 - 3:

1. $N_0 := \emptyset$ ja $i := 1$;
2. $N_1 := N_{i-1} \cup \{A \mid A \longrightarrow \alpha \in P,\ \alpha \in (N_{i-1} \cup \Sigma)^*\}$;
3. Kui $N_i \neq N_{i-1}$, siis $i := i + 1$ ja jätkata p.2.;
4. Kui $S \in N_e$, siis **stop** ("JAH"), vastasel juhul **stop** ("EI").

□

Algoritmi 1 ajaline keerukus on $\mathcal{O}(n)$, kus $n = \mid P \mid$.

Sümbolit $x \in N \cup \Sigma$ nimetame *saavutamatuks* KV-grammatikas G, kui ei leidu tuletatavat lausevormi $\alpha = \beta X \gamma$, s.t. $\neg(S \Longrightarrow_G^* \beta X \gamma)$.

Algoritm 2. (Saavutamatute sümbolite elimineerimine).
Sisend: KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: KV-grammatika $G' = (\Sigma', N', P', S)$, nii et

- $\mathcal{L}(G') = \mathcal{L}(G)$;
- iga $X \in N' \cup \Sigma'$ korral leiduvad stringid $\alpha, \beta \in (N' \cup \Sigma')^*$, nii et $S \Longrightarrow_G^* \alpha X \beta$.

Meetod:

1. $V_0 := \{S\}$ ja $i := 1$;
2. $V_i := \{X \mid A \longrightarrow \alpha \in P,\ A \in V_{i-1}\} \cup V_{i-1}$;
3. kui $V_1 \neq V_{i-1}$, siis $i := i + 1$ ja jätkata p. 2,
   vastasel juhul

   $N' := V_i \cap N$;
   $\Sigma' := V_i \cap \Sigma$;
   $P' := \{A \longrightarrow \alpha \mid A \longrightarrow \alpha \in P,\ \ A \in N',\ \ \alpha \in (N' \cup \Sigma')^*\}$.
   $G' := (\Sigma', N', P', S)$.;
4. **stop** (G').

□

Algoritmi 2 ajaline keerukus on $\mathcal{O}(n)$.

Sümbolit $X \in N \cup \Sigma$ nimetame *kasutuks* KV-grammatikas $G = (\Sigma, N, P, S)$, kui ei leidu tuletust

$$S \Longrightarrow_G^* wXy \Longrightarrow_G^* wxy,$$

kus $w, x, y \in \Sigma^*$.

Algoritm 3. (Kasutute sümbolite elimineerimine).
Sisend: KV-grammatika $G = (\Sigma, N, P, S)$, nii et $\mathcal{L}(G) \neq \emptyset$.
Väljund: KV-grammatika $G' = (\Sigma', N', P', S)$, nii et $\mathcal{L}(G') = \mathcal{L}(G)$ ja hulk $\Sigma' \cup N'$ ei sisalda kasutuid sümboleid.
Meetod:

1. Kasutades algoritmi 1 konstrueerida hulk N_e ning $G_1 := (N \cap N_e, \Sigma, P_1, S)$, kus

$$P_1 = \{A \longrightarrow \alpha \mid A \longrightarrow \alpha \in P,\ \ \alpha \in (N_e \cup \Sigma)^*\};$$

2. Rakendades grammatikale G_1 algoritmi 2, konstrueerida $G' = (\Sigma', N', P', S)$;
3. **stop** (G').

□

Näide 23. Vaatleme KV-grammatikat produktsioonidega

$$S \longrightarrow a$$
$$S \longrightarrow A$$
$$A \longrightarrow AB$$
$$B \longrightarrow b$$

Kuna antud juhul $N = \{S, B\}$, siis algoritmi 3 p.1 tulemuseks on grammatika

$$G_2 = (\{a, b\},\ \{S, B\},\ \{S \longrightarrow a,\ B \longrightarrow b\},\ S).$$

Rakendades viimasele algoritmi 2 saame

$$V_2 = V_1 = \{S, a\}$$

ning

$$G' = (\{a\},\ \{S\},\ \{S \longrightarrow a\},\ S).$$

□

ϵ-reegliks nimetame produktsiooni kujul $A \longrightarrow \epsilon$.

KV-grammatikat $G = (\Sigma, N, P, S)$ nimetatakse ϵ-vabaks, kui
1. P ei sisalda ϵ-reegleid või
2. P sisaldab parajasti ühe S-reegli $S \longrightarrow \epsilon$ ning lähtesümbol S ei sisaldu ühegi produktsiooni paremas pooles.

Algoritm 4. (ϵ-reeglite elimineerimine).
Sisend: KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: Grammatikaga G ekvivalentne KV-grammatika $G' = (\Sigma, N', P', S)$, milles pole ϵ-reegleid.
Meetod:

1. Analoogselt algoritmidele 1 ja 2 konstrueerida hulk $N_e = \{A \mid A \Longrightarrow_G^* \epsilon\}$;
2. Konstrueerida hulk P' järgmiselt:
   (a) Kui $A \longrightarrow \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_k \alpha_k \in P$ $k \geq 0$ ja $B_i \in N_e$ iga $1 \leq i \leq k$ korral, kuid ükski stringide $\alpha_j (0 \leq j \leq k)$ sümbol ei kuulu hulka N_e, siis lülitada hulga P' koosseisu kõik reeglid kujul

   $$A \longrightarrow \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \ldots X_k \alpha_k,$$

   kus X_i on kas B_i või ϵ, väljaarvatud produktsioon $A \longrightarrow \epsilon$;
   (b) Kui $S \in N_e$, siis lülitada hulga P' koosseisu produktsioonid

   $$S' \longrightarrow S$$
   $$S' \longrightarrow \epsilon,$$

   kus S' on uus mitteterminaal (s.t. $N' := N \cup \{S'\}$), vastasel korral $N' := N,\ \ S' := S$;
3. $G' = (\Sigma, N', P', S')$;
4. **stop** (G').

□

Näide 24. Rakendades algoritmi 4 grammatikale

$$S \longrightarrow aSbS$$
$$S \longrightarrow bSaS$$
$$S \longrightarrow \epsilon,$$

saame tulemuseks:

$$\begin{array}{ll} S' \longrightarrow \epsilon & S \longrightarrow abS \\ S' \longrightarrow S & S \longrightarrow ab \\ S \longrightarrow aSbS & S \longrightarrow bSa \\ S \longrightarrow bSaS & S \longrightarrow baS \\ S \longrightarrow aSb & S \longrightarrow ba \end{array}$$

□

Algoritm 5. (Ahelproduktsioonide elimineerimine).
Sisend: ϵ-vaba KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: Grammatikaga G ekvivalentne KV-grammatika $G' = (\Sigma, N, P', S)$, milles pole ahelproduktsioone.
Meetod:

1. Iga $A \in N$ jaoks konstrueerida hulk $N_A = \{B \mid A \Longrightarrow^* B\}$ järgmisel viisil:
   (a) $N_0 := \{A\}$ ja $i := 1$;
   (b) $N_i : +N_{i-1} \cup \{C \mid B \longrightarrow C \in P, \;\; B \in N_{i-1}\}$;
   (c) kui $N_i \neq N_{i-1}$, siis $i := +1$ ja jätkata p. (b),
   vastasel juhul $N_A := N_i$;
2. Konstrueerida hulk P' järgmiselt: kui $B \longrightarrow \alpha \in P$ pole ahelproduktsioon, lülitada hulga P' koosseisu produktsioon $A \longrightarrow \alpha$ kõigi selliste $A \in N$ jaoks, et $B \in N_A$;
3. $G' = (\Sigma, N, P', S')$;
4. **stop** (G').

□

Näide 25. Vaatleme KV-grammatikat aritmeetiliste avaldiste genereerimiseks:

$$\begin{array}{lll} E \longrightarrow E + T & T \longrightarrow T * F & F \longrightarrow (E) \\ E \longrightarrow T & T \longrightarrow F & F \longrightarrow a \end{array}$$

Rakendades sellele grammatikale algoritmi 5, konstrueerime kõigepealt hulgad $N_E = \{E, T, F\}$, $N_T = \{T, F\}$ ja $N_F = \{F\}$ ning teisendame seejärel grammatika kujule

$$\begin{array}{lll} E \longrightarrow E + T & T \longrightarrow T * F & F \longrightarrow (E) \\ E \longrightarrow T * F & T \longrightarrow (E) & F \longrightarrow a \\ E \longrightarrow (E) & T \longrightarrow a & \\ E \longrightarrow a & & \end{array}$$

□

Definitsioon 41. *KV-grammatikat $G = (\Sigma, N, P, S)$ nimetatakse* redutseerituks, *kui ta ei sisalda tsükleid (ei saa moodustada tuletusi $A \Longrightarrow^+ A, \;\; A \in N$), ϵ-reegleid ja kasutuid sümboleid.*

Teoreem 11. *Iga KV-keele L jaoks leidub selline redutseritud KV-grammatika G, nii et $\mathcal{L}(G) = L$.*

Iga KV-grammatika G on redutseeritav, kasutades algoritme 1 - 4.

3.6.3. KV-grammatikate normaalkujud

Olgu KV-grammatikas $G = (\Sigma, N, P, S)$ produktsioon $A \longrightarrow \alpha B \beta$, kus $B \in N$ ja $\alpha, \beta \in (\Sigma \cup N)^*$. Lisaks eeldame, et hulk P sisaldab produktsioonid $B \longrightarrow \gamma_1$, $B \longrightarrow \gamma_2, \ldots, B \longrightarrow \gamma_k$, mille vasak pool on mitteterminaal B. Moodustame uue KV-grammatika $G' = (\Sigma, N, P', S)$, kus

$$P' = (P \backslash \{A \longrightarrow \alpha B \beta\}) \cup \{A \longrightarrow \alpha\gamma_1\beta, \; A \longrightarrow \alpha\gamma_2\beta, \ldots, \; A \longrightarrow \alpha\gamma_k\beta\}.$$

Lemma 3. $\mathcal{L}(G) = \mathcal{L}(G')$.

Tõestus. Olgu grammatikas G tuletus, milles kasutatakse produktsiooni $A \longrightarrow \alpha B \beta$:

$$S \Longrightarrow^* x_1 A y_1 \Longrightarrow x_1 \alpha B \beta y_1 \Longrightarrow^* x_2 B y_2 \Longrightarrow x_2 \gamma_i y_2,$$

siis on grammatikas G' moodustatav tuletus

$$S \Longrightarrow^* x_1 A y_1 \Longrightarrow x_1 \alpha \gamma_i \beta y_1 \Longrightarrow^* x_2 \gamma_i y_2.$$

Vastupidine tuletustevaheline teisendus on analoogiline.

□

Definitsioon 42. *KV-grammatika $G = (\Sigma, N, P, S)$ öeldakse olevat Chomsky normaalkujul (binaarsel normaalkujul), kui iga produktsioon hulgas P on ühel järgmistest kujudest:*

1. *$A \longrightarrow BC$, kus $A, B, C \in N$;*
2. *$A \longrightarrow a$, kus $a \in \Sigma$;*
3. *$S \longrightarrow \epsilon$, kui $\epsilon \in \mathcal{L}(G)$ ja grammatika lähtesümbol S ei esine produktsioonide paremal poolel.*

Algoritm 6. (Grammatika teisendamine Chomsky normaalkujule).
Sisend: Redutseeritud KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: KV-grammatika $G' = (\Sigma, N'P', S)$ Chomsky normaalkujul, kusjuures

$$\mathcal{L}(G) = \mathcal{L}(G').$$

Meetod:

1. $P' := \{A \longrightarrow a \mid A \longrightarrow a \in P,\ a \in \Sigma\} \cup \{A \longrightarrow BC \mid A \longrightarrow BC \in P,$
   $B\ C \in N\} \cup \{S \longrightarrow \epsilon \mid S \longrightarrow \epsilon \in P\}$;
2. Iga produktsiooni $A \longrightarrow X_1 \dots X_k \in P$ korral, kus $k > 2$ täiendada hulka P':
   $P' := P' \cup \{A \longrightarrow X_1' < X_2 \dots X_k >, < X_2 \dots X_k > \longrightarrow X_2' < X_3 \dots X_k >, \dots,$
   $< X_{k-1} X_k > \longrightarrow X_{k-1}' X_k'\}$, kus $X_i' = X_i$, kui $X_i \in N$ ja X_i - uus mitteterminaal, kui $X_i \in \Sigma$, $< X_i \dots X_k >$ on uus mitteterminaal;
3. $P' := P' \cup \{A \longrightarrow X_1' X_2' \mid A \longrightarrow X_1 X_2 \in P$ ja $X_1 \in P$ või $X_2 \in \Sigma$,
   kusjuures X_1' ja X_2' omavad sama tähendust nagu p.2;
4. Iga uue mitteterminaali a' jaoks, mis toodi sisse sammude 2 ja 3 täitmisel, lisada hulka P' produktsioon $a' \longrightarrow a$;
5. Konstrueerida hulk N', lisades hulgale N kõik sammude 2 ja 3 täitmisel lisatud uued mitteterminaalid ja $G' = (\Sigma, N', P', S)$;
6. **stop** (G').

□

Näide 26. teisendades Chomsky normaalkujule KV-grammatika

$$\begin{array}{ll} S \longrightarrow aAB & A \longrightarrow a \\ S \longrightarrow BA & B \longrightarrow AS \\ A \longrightarrow BBB & B \longrightarrow b \end{array}$$

Kasutusele võtame uued mitteterminaalid:

A_1 - sümboli a tähistamiseks produktsioonis $S \longrightarrow aAB$;
B_1 - alamstringi AB tähistamiseks;
B_2 - alamstringi BB tähistamiseks.

Tulemuseks saame grammatika

$$
\begin{array}{lll}
S \longrightarrow A_1B_1 & A \longrightarrow a & A_1 \longrightarrow a \\
S \longrightarrow BA & B \longrightarrow AS & B_1 \longrightarrow AB \\
A \longrightarrow BB_2 & B \longrightarrow b & B_2 \longrightarrow BB
\end{array}
$$

□

Teoreem 12. *Iga KV-grammatika G on teisendatav temaga ekvivalentseks grammatikaks G' Chomsky normaalkujul, kasutades algoritmi 6.*

Tõestus tugineb otseselt lemmale 1.

Definitsioon 43. *ϵ-vaba KV-grammatika $G = (\Sigma, N, P, S)$ on Greibachi normaalkujul, kui kõik produktsioonid (v.a. $S \longrightarrow \epsilon$, kui selline esineb hulgas P) on kujul $A \longrightarrow a\alpha$, kus $a \in \Sigma$ ja $\alpha \in N^*$.*

Mitteterminaali A nimetame rekursiivseks, kui $A \Longrightarrow^+ \alpha A\beta$. Kui $\alpha = \epsilon$, siis A on vasakrekursiivne, kui $\beta = \epsilon$, siis A on paremrekursiivne.

Iga produktsiooni kujul $A \longrightarrow \alpha$ nimetame A-produktsiooniks.

Lemma 4. *Olgu $G = (\Sigma, N, P, S)$ KV-grammatika, mille kõikideks A-produktsioonideks on*

$$
\begin{array}{l}
A \longrightarrow A\alpha_1 \\
A \longrightarrow A\alpha_2 \\
\quad \vdots \\
A \longrightarrow A\alpha_m \\
A \longrightarrow \beta_1 \\
A \longrightarrow \beta_2 \\
\quad \vdots \\
A \longrightarrow \beta_n,
\end{array}
$$

kus mitte ükski β_i-dest ei alga A-ga.

Olgu $G' = (\Sigma, N \cup \{A'\}, P', S)$, kus A' on uus mitteterminaal ning P' on saadud hulgast P kõigi A-produktsioonide asendamisel produktsioonidega

$$
\begin{array}{ll}
A \longrightarrow \beta_1 & A' \longrightarrow \alpha_1 \\
\quad \vdots & \quad \vdots \\
A \longrightarrow \beta_n & A' \longrightarrow \alpha_m \\
A \longrightarrow \beta_1 A\,' & A' \longrightarrow \alpha_1 A' \\
\quad \vdots & \quad \vdots \\
A \longrightarrow \beta_n A' & A' \longrightarrow \alpha_m A'.
\end{array}
$$

Siis $\mathcal{L}(G') = \mathcal{L}(G)$.

Tõestus. Kasutades grammatikas G vasaktuletust, saab A-produktsioonide alusel tuletada regulaarse avaldisega $(\beta)1 + \ldots + \beta_n)(\alpha_1 + \ldots + \alpha_m)^*$ määratud lausevorme. Sama hulga saab tuletada grammatikas G', kasutades algulüht A-produktsiooni ning seejärel teatud arv kordi A'-produktsiooni. Seega $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. Vastupidine sisalduvus tõestatakse analoogiliselt, lähtudes paremtuletusest grammatikas G'.

□

Algoritm 7. (Vasakrekursiooni elimineerimine)
Sisend: Redutseeritud KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: Ekvivalentne KV-grammatika $G' = \Sigma, N', P', S)$, milles pole vasakrekursiivseid produktsioone.

Meetod:

1. Olgu $N = \{A_1, \ldots, A_n\}$. Teisendame grammatikat nii, et produktsioonis $A_1 \longrightarrow \alpha$ algaks string α kas terminaaliga või mitteterminaaliga A_j, kusjuures $j > i$. Selleks $i := 1$;
2. Olgu A_i-produktsioonide hulk grammatikas G

$$\begin{array}{l} A_i \longrightarrow A_i\alpha_1 \\ \quad\vdots \\ A_i \longrightarrow A_i\alpha_m \\ A_i \longrightarrow \beta_1 \\ \quad\vdots \\ A_i \longrightarrow \beta_p, \end{array}$$

kus üks stringidest β_j algab A_k-ga, kui $k \leq i$. Asendame need reeglid uute reeglitega, tuues ühtlasi sisse ka uue mitteterminaali A':

$$\begin{array}{ll} A_i \longrightarrow \beta_1 & A_i' \longrightarrow \alpha_1 \\ \quad\vdots & \quad\vdots \\ A_i \longrightarrow \beta_p & A_i' \longrightarrow \alpha_m \\ A_i \longrightarrow \beta_1 A_i' & A_i' \longrightarrow \alpha_1 A_i' \\ \quad\vdots & \quad\vdots \\ A_i \longrightarrow \beta_p A_i' & A_i' \longrightarrow \alpha_m A_i'; \end{array}$$

3. Kui $i = n$, siis **stop** (G'), vastasel juhul $i := i + 1,\ j := 1$;
4. Asendada iga reegel $A_i \longrightarrow A_j\alpha$ reeglitega

$$\begin{array}{l} A_i \longrightarrow \beta_1\alpha \\ \quad\vdots \\ A_i \longrightarrow \beta_m\alpha, \end{array}$$

kus $A_j \longrightarrow \beta_1, \ldots, A_j \longrightarrow \beta_m$ on grammatika G kõik A_j-reeglid;
5. Kui $j = i - 1$, jätkata p. 2, vastasel juhul $j := j + 1$ ja jätkata p. 4.

□

Teoreem 13. *Iga KV-keel on genereeritav mitte-vasakrekursiivse grammatikaga.*

Tõestus. Piisab tõestada, et algoritm 7 tulemus on mitte-vasakrekursiivne grammatika. ($\mathcal{L}(G) = \mathcal{L}(G')$, kuna algoritmis 7 kasutatakse lemmades 1 ja 2 vaadeldud teisendusi). Püstitame kaks väidet:

A. Pärast sammu 2 algab iga A_i-produktsiooni parem pool terminaaliga või mitteterminaaliga A_k, kus $k > i$.
B. Pärast sammu 4 algab iga A_i-produktsiooni parem pool terminaaliga või mitteterminaaliga A_k, kus $k > j$.

Näitame, et väide A kehtib kõikide $i \leq n$ korral (sellest piisab ka kogu teoreemi tõestuseks) ja väide B kehtib kõikide paaride (i, j) jaoks, kus $j < i \leq n$ ja $i.j$ (vajalik väite A tõestamiseks).

Väidete A ja B parameetrid i ja (i, j) esinevad algoritmi töö käigus järgmise korra järgi:
1, (2, 1), 2, (3, 1), (3, 2), 3, (4, 1), (4, 2), (4, 3), 5, (5, 1), ..., (n, n-1), n.
Tõestame väited A ja B induktsiooniga selle rea järgi.

Baas: $i = 1$ korral on väide A tõene, kuna sümboliga A_1 ei saa alata ükski stringidest $\beta_1, \ldots, \beta_p$.
Samm: Oletame, et väited A ja B on tõesed parameetrile (i, j) eelnevate väärtuste korral. Kuna $j < i$, siis induktsiooni eelduse põhjal on väide A tõene j jaoks. Seega sammul 4 algavad $\beta_1, \ldots, \beta_m$ terminaalidega või A_k, kus $k > j$. Pärast sammu 4 lõppemist on $\beta_1, \ldots, \beta_m$ A_i-produktsioonide prefiksiteks. Seega on B tõene paari (i, j) jaoks.

Kui eeldada, et A ja B on tõesed iga parameetri jaoks, mis eelneb i-le, saab analoogselt näidata, et ka väide A kehtib i korral.

Niisiis väitest A järeldub, et $A_1, A_2, \ldots, A_n$ pole vasakrekursiivsed. Samuti pole vasakrekursiivsed sammul 2 sissetoodud sümbolid A_i'. Viimane järeldub sellest, et grammatika G redutseerimise tõttu pole ükski $\alpha_1, \ldots, \alpha_m$ tühi sõna.

□

Näide 27. Vaatleme algoritmi 7 tööd grammatika

$$\begin{array}{lll} A \longrightarrow BC & A \longrightarrow a & \\ B \longrightarrow CA & B \longrightarrow Ab & \\ C \longrightarrow AB & C \longrightarrow CC & C \longrightarrow a \end{array}$$

korral.

Kui valida $A_1 = A$, $A_2 = B$ ja $A_3 = C$, on algoritmi 7 sammude 2 ja 4 järel grammatika järgmine (kirjas on vaid muudetavad produktsioonid):

Samm 2, $i = 1$: grammatika ei muutu
Samm 4, $i = 2$, $j = 1$: $B \longrightarrow CA$, $B \longrightarrow BCb$, $B \longrightarrow ab$
Samm 2, $i = 2$: $B \longrightarrow CA$, $B \longrightarrow ab$, $B \longrightarrow CAB'$, $B \longrightarrow abB'$, $B' \longrightarrow Cb$, $B' \longrightarrow CbB'$
Samm 4, $i = 3$, $j = 1$: $C \longrightarrow BCB$, $C \longrightarrow aB$, $C \longrightarrow CC$, $C \longrightarrow a$
Samm 4, $i = 3$, $j = 2$: $C \longrightarrow CACB$, $C \longrightarrow abCB$, $C \longrightarrow CAB'CB$, $C \longrightarrow abB'CB$, $C \longrightarrow aB$, $C \longrightarrow CC$, $C \longrightarrow a$
Samm 2, $i = 3$: $C \longrightarrow abCB$, $C \longrightarrow abB'CB$, $C \longrightarrow aB$, $C \longrightarrow a$, $C \longrightarrow abCBC'$, $C \longrightarrow abB'CBC'$, $C \longrightarrow aBC'$, $C \longrightarrow aC'$, $C' \longrightarrow ACBC'$, $C' \longrightarrow AB'CBC'$, $C \longrightarrow CC'$, $C' \longrightarrow ACB$, $C' \longrightarrow AB'CB$, $C' \longrightarrow C$

□

Näide 28. Aritmeetilise avaldise grammatika

$$E \longrightarrow E + T,\ E \longrightarrow T,\ T \longrightarrow T * F,\ T \longrightarrow F,\ F \longrightarrow a,\ F \longrightarrow (E)$$

korral annab algoritm 7 tulemuseks

$$\begin{array}{ll} E \longrightarrow T & E \longrightarrow TE' \\ E' \longrightarrow +T & E' \longrightarrow +TE' \\ T \longrightarrow F & T \longrightarrow FT' \\ T' \longrightarrow *F & T \longrightarrow *FT' \\ F \longrightarrow (E) & F \longrightarrow a. \end{array}$$

□

Lemma 5. *Kui KV-grammatika $G = (\Sigma, N, P, S)$ on mittevasakrekursiivne, siis leidub mittetrminaalide hulgal N järjestus, nii et $A \longrightarrow B\ \alpha \in P \Longrightarrow A < B$.*

Algoritm 8. (KV-grammatika viimine Greibachi normaalkujule).
Sisend: Mitte-vasakrekursiivne KV-grammatika $G = (\Sigma, N, P, S)$.
Väljund: Grammatikaga G ekvivalentne KV-grammatika G' Greibachi normaalkujul.

Meetod:

1. Konstrueerida vastavalt lemma 5 mitteterminaalide hulgal N selline järjestus $<$, nii et iga A-produktsiooni parem pool algab terminaaliga või mitteterminaaliga B, kusjuures $A < B$;
2. $i := n - 1$;
3. Kui $i = 0$, siis samm 5,
   vastasel juhul asendada produktsioon $A_i \longrightarrow A_j\alpha$, kus $j > i$, reeglitega
   $$\begin{aligned} A_i &\longrightarrow \beta_1\alpha \\ &\vdots \\ A_i &\longrightarrow \beta_m\alpha, \end{aligned}$$
   kus $A_j \longrightarrow \beta_1, \ldots, A_j \longrightarrow \beta_m$ on kõik A-produktsioonid grammatikas G. (Lihtne on näha, et $\beta_1, \ldots, \beta_m$ algavad terminaaliga);
4. $i := i - 1$ ja pöörduda tagasi p. 3 juurde;
5. Nüüd algab kõigi produktsioonide parem pool (v. a. $S \longrightarrow \epsilon$, kui selline üldse esineb keeles) terminaaliga.
   Igas produktsioonis $A \longrightarrow aX_1 \ldots X_n$ asendada $X_j \in \Sigma$ uue mitteterminaaliga X_j';
6. Iga eelmisel sammul lisatud mitteterminaali X_j' jaoks lisada produktsioonid $X_j' \longrightarrow X_j$.

□

Teoreem 14. *Iga keele L jaoks leidub grammatika G Greibachi normaalkujul, nii et $\mathcal{L}(G) = L$.*

Näide 29. Rakendame algoritmi 8 näite 28 tulemusele, valides järjestuse

$$E' < E < T' < T < F.$$

Saadav grammatika oleks kujul

$E \longrightarrow (EE_1$	$T \longrightarrow (EE_1$
$E \longrightarrow a$	$T \longrightarrow a$
$E \longrightarrow (EE_1T'$	$T \longrightarrow EE_1T'$
$E \longrightarrow aT'$	$T \longrightarrow aT'$
$E \longrightarrow (EE_1E'$	$T' \longrightarrow *F$
$E \longrightarrow aE'$	$T' \longrightarrow *FT'$
$E \longrightarrow (EE_1T'E$	$F \longrightarrow EE_1$
$E \longrightarrow aT'E'$	$E \longrightarrow a$
$E' \longrightarrow +T$	$E_1 \longrightarrow)$
$E' \longrightarrow +TE'$	

□

3.7. Kontekstivabade keelte süntaksanalüüsi algoritmid

Süntaksanalüüsi ülesande korral on antud keelt genereeriva grammatika $G = (\Sigma, N, P, S)$ põhjal tarvis otsustada, kas sõna w kuulub keelde $L(G)$ või mitte. Järgnevalt vaatlemegi mõningaid algoritme nimetatud ülesande lahendamiseks KV-grammatikate korral.

3.7.1. Earley' algoritm

Earley' algoritm püüab süntaksanalüüsi ülesande lahendamiseks ehitada sõna w vasaktuletust. Grammatika G produktsiooni $A \longrightarrow v$ võib olla kasutatud sõna $w = x_1 \dots x_n$, kus $x_j \in \Sigma$, vasaktuletuses vaid siis, kui leidub tuletus

$$S \Longrightarrow^* x_1 \dots x_i A\delta.$$

Oletame, et sõn v koosneb alamsõnadest α ja β, s.t. $v = \alpha\beta$. Sellise täpsustuse korral saab produktsiooni $A \longrightarrow v$ kasutatavuse tervikliku tingimuse esitada täpsemalt: antud juhul peab leiduma tuletus

$$S \Longrightarrow^* x_1 \dots x_i A\delta \Longrightarrow x_1 \dots x_i\alpha\beta\delta \Longrightarrow^* x_1 \dots x_i x_{i+1} \dots x_j\beta\delta. \tag{13}$$

Paneme tähele, et sellise tuletuse olemasolu pole piisav, et sõna w oleks tervikuna tuletatav, sest terminaalse stringi $x_{j+1} \dots x_n$ tuletatavus lausevormist $\alpha\beta$ pole garanteeritud.

Earley' algoritmis moodustatakse massiiv $\|I_{ij}\|$, mille elementideks on nn. punktiga produktsioonide hulgad. Nimelt kuulub produktsioon $A \longrightarrow \alpha.\beta$ elemendi I_{ij} koosseisu parajasti siis, kui sõna $w = x_1 \dots x_n$ vasaktuletuse võib ehitada osatuletuse (13) jätkamise teel. Sõna w kuulub keelde $\mathcal{L}(G)$, kui produktsioon $S \longrightarrow v.$ kuulub elementi $I_{0,n}$.

Algoritm 9.
Antud: ϵ-vaba KV-grammatika $G = (\Sigma, N, P, S)$
ja string $w = x_1 \dots x_n \in \Sigma^*$
Tulemus: maatriks $\|Iij\|$, kus $0 \leq i \leq j \leq n,\ i < n;\ w \in \mathcal{L}(G)$, kui leidub produktsioon $S \longrightarrow v \in P$ ja $S \longrightarrow v. \in I_{0,n}$.
Meetod:

1. *Initsialiseerimine*: iga produktsiooni $S \longrightarrow v \in P$ jaoks lisada hulka $I_{0,0}$ element $S \longrightarrow .v$;
2. *Laiendamine*: niikaua kui hulka $I_{0,0}$ lisandub uusi elemente, teostada operatsioon: "Iga sellise $A \in N$ jaoks, mille korral $B \longrightarrow .Au \in I_{0,0}$ ja $A \longrightarrow v \in P$, lisada $I_{0,0}$-sse element $A \longrightarrow .v$";
3. Iga $j > 0$ korral korrata samme (a), (b) ja (c), kuni vähemalt ühte elementidest $I_{0j}, I_{1j}, \dots, I_{jj}$ lisandub uusi punktiga produktsioone;
   (a) *Nihe*: Kui $A \longrightarrow w_1.x_j w_2 \in I_{i,j-1}$, siis lisada elementi I_{ij} produktsioon $A \longrightarrow w_1 x_j.w_2$;
   (b) *Laiendamine*: kui $A \longrightarrow w_1.Bw_2 \in I_{ij}$ ja $B \longrightarrow v \in P$, siis lisada elementi I_{jj} produktsioon $B \longrightarrow .v$;
   (c) *Taandamine*: Kui $A \longrightarrow w. \in I_{ij}$ ja $B \longrightarrow w_1.Aw_2 \in I_{ki}$, siis lisada elementi I_{kj} produktsioon $B \longrightarrow w_1 A.w_2$.

□

Näide 30. Vaatleme KV-grammatikat, mis genereerib muutujate a ja b abil moodustatud võrdused $a = b$, $(a = (a + b))$ jne.

$$\begin{array}{ll} S \longrightarrow (S) & E \longrightarrow (E + E) \\ S \longrightarrow R & E \longrightarrow a \\ R \longrightarrow E = E & E \longrightarrow b \end{array}$$

Näiteks stringi $a = b$ analüüsi käigus koostatakse Earley' algoritmi alusel järgmine tabel $\|I_{ij}\|$:

$i \setminus j$	0	1	2	3
0	$S \longrightarrow .(S)$ $S \longrightarrow .R$ $R \longrightarrow .E = E$ $E \longrightarrow .(E + E)$ $E \longrightarrow .a$ $E \longrightarrow .b$	$R \longrightarrow E. = E$ $E \longrightarrow a.$	$R \longrightarrow E = .E$	$S \longrightarrow R.$ $R \longrightarrow E = E.$
1				
2			$E \longrightarrow .(E + E)$ $E \longrightarrow .a$ $E \longrightarrow .b$	$E \longrightarrow b.$
		a	=	b

String kuulub vaadeldava grammatika poolt genereeritavasse keelde, kuna element $I_{0,3}$ sisaldab produktsiooni $S \longrightarrow R.$.

□

Ülesanne 2. *Koostada Earley' algoritmi "olekute tabel" näites 30 esitatud grammatika ja lähtesõna $(a + b) = b$ korral.*

Lemma 6. *Olgu $B \longrightarrow .v \in I_{ii}$ ning eeldame, et $v \Longrightarrow^* x_{i+1} \dots x_j$. Siis $B \longrightarrow v. \in I_{ij}$.*

Tõestus. Tõestame lemma induktsiooniga sõnast v tuletatava stringi pikkuse $k = j - i$ järgi.

Kui $k = 1$, siis on kaks võimalust:

1. $B \longrightarrow .x_j \in I_{ii}$ ning algoritmi 9 sammu (a) põhjal $B \longrightarrow x_j. \in I_{ij}$;
2. $B \Longrightarrow C_1 \Longrightarrow C_2 \Longrightarrow \dots \Longrightarrow C_m \Longrightarrow x_j$, kus $C_1 = v$.

Sel juhul kuuluvad sammu (b) põhjal elemendi I_{ii} koosseisu ka produktsioonid

$$\begin{array}{ll} B & \longrightarrow .\, C_1 \\ C_1 & \longrightarrow .\, C_2 \\ & \vdots \\ C_{m-1} & \longrightarrow .\, C_m \\ C_m & \longrightarrow .\, x_j \end{array}$$

Sammu (c) korduva rakendamise tulemusena lisatakse elementi I_{ij} produktsioonid

$$C_m \longrightarrow x_j., C_{m-1} \longrightarrow C_m., \dots, B \longrightarrow C_1.$$

Oletame, et tulemus kehtib iga $j - i < k$ korral. Vaatleme juhtu $j - i = k$.

Kui v koosneb ühest sümbolist, toimub ahelproduktsioonide asendamine analoogselt juhtumiga 2 ülal. Seepärast võime eeldada, et $v = z_1 \dots z_m$, kus $m > 1$ ja iga $z_s \in \Sigma \cup N$.

Siis on võimalikud kaks juhtu:

1. $z_1 = x_{i+1}$ ja $B \longrightarrow x_{i+1}.z_2 \dots z_m \in I_{i,i+1}$ sammu (a) tõttu;
2. $z_1 = C \in N$ ja $C \Longrightarrow v' \Longrightarrow^* x_{i+1} \dots x_q$. Siis $C \longrightarrow .v' \in I_{ii}$ ja $C \longrightarrow v'. \in I_{iq}$ induktsiooni eelduse tõttu. Siis kuulub aga sammu (c) tõttu elemendi I_{iq} koosseisu ka produktsioon $B \longrightarrow C.z_2 \dots z_m$.

Analoogiliselt saame toimida ka $z_2 \dots z_m$ korral ning tuletada, et $B \longrightarrow v. \in I_{ij}$.

□

Lemma 7. *Olgu $A \longrightarrow w_1.w_2w_3 \in I_{ik}$, kus $w_1, w_2, w_3 \in (\Sigma \cup N)^*$ ja oletame, et $w_2 \Longrightarrow^* x_{k+1} \dots x_j$. siis $A \longrightarrow w_1w_2.w_3 \in I_{ij}$.*

Tõestus. Induktsioon sõna w_2 pikkuse järgi.
Kui $\mid w_2 \mid = 0$, kehtib lemma triviaalselt.

Oletame, et lemma kehtib iga $\mid w_2 \mid < p$ korral ning vaatleme juhtu $\mid w_2 \mid = p > 0$.
Kui $w_2 = x_{k+1}w_2'$, siis sammu (a) tõttu

$$A \longrightarrow w_1 x_{k+1}.w_2' w_3 \in I_{i,k+1}.$$

Kuna $w_2' \Longrightarrow^* x_{k+2} \ldots x_j$ ja $\mid w_2' \mid < p$, siis induktsiooni eelduse põhjal:

$$A \longrightarrow w_1 w_2.w_3 \in I_{ij}.$$

Juhul kui $w_2 = Bw_2'$ ning $B \Longrightarrow u \Longrightarrow^* x_{k+1} \ldots x_m$, kehtib sammu (b) tõttu tingimus $B \longrightarrow .u \in I_{kk}$. Lemma 6 tõttu $B \longrightarrow u. \in I_{km}$ ning sammu (c) tulemusena $A \longrightarrow w_1 B.w_2' w_3 \in I_{im}$. Kasutades induktsiooni eeldust, saame $A \longrightarrow w_1 Bw_2.'w_3 \in I_{ij}$.

□

Lemma 8. *Olgu $B \Longrightarrow w \Longrightarrow^* w_1 A w_2$, kus $w, w_1, w_2 \in (N \cup \Sigma)^*$ ning $B \longrightarrow .w \in I_{kk}$ ja $w \Longrightarrow x_{k+2} \ldots x_j$. Siis iga A-produktsiooni $A \longrightarrow Z$ korral $A \longrightarrow .Z \in I_{jj}$.*

Tõestus. (Harjutusülesanne)

□

Teoreem 15. *(Earley' algoritmi korrektsus). Etteantud stringi $w = x_1 \ldots x_n \in \Sigma^*$ korral paigutub algoritm 9 produktsiooni $A \longrightarrow \alpha.\beta$ elementi I_{ij} parajasti siis, kui grammatikas leidub vasaktuletus (13):*

$$S \Longrightarrow^* x_1 \ldots x_i A\delta \Longrightarrow x_1 \ldots x_i \alpha\beta\delta \Longrightarrow^* x_i \ldots x_j \beta\delta.$$

Erijuhul, kui $S \longrightarrow v. \in I_{0,n}$, siis $w \in \mathcal{L}(G)$.

Tõestus.
Tarvilikkus. Kui algoritm 9 paigutab $A \longrightarrow \alpha.\beta$ elementi I_{ij}, siis on rahuldatud (13) - tõestada harjutusülesandena iseseisvalt.
Piisavus. Formuleerime tingimuse (13) ümber: leiduvad stringid $w_1, w_2 \in (\Sigma \cup N)^*$ ning produktsioon $A \longrightarrow w_1 w_2$, nii et

1. $S \Longrightarrow v \Longrightarrow^* w_3 A\delta$
2. $w_3 \Longrightarrow^* a_1 \ldots a_i$
3. $w_1 \Longrightarrow^* a_{i+1} \ldots a_j$

Soovime näidata, et sel korral $A \longrightarrow w_1.w_2 \in I_{ij}$. Algoritmi 9 sammu 1 tõttu $S \longrightarrow .v \in I_{00}$ ja siis järeldub lemma 8 alusel, et $A \longrightarrow .w_1 w_2 \in I_{ii}$. Lemma 7 tõttu kehtib siis ka $A \longrightarrow w_1.w_2 \in I_{ij}$.

□

Teoreem 16. *Earley' algoritmi ajaline keerukus on $\mathcal{O}(n^3)$, kus n on analüüsitava sõna pikkus; juhul kui KV-grammatika G on ühene, on ajaline keerukus $\mathcal{O}(n^2)$.*

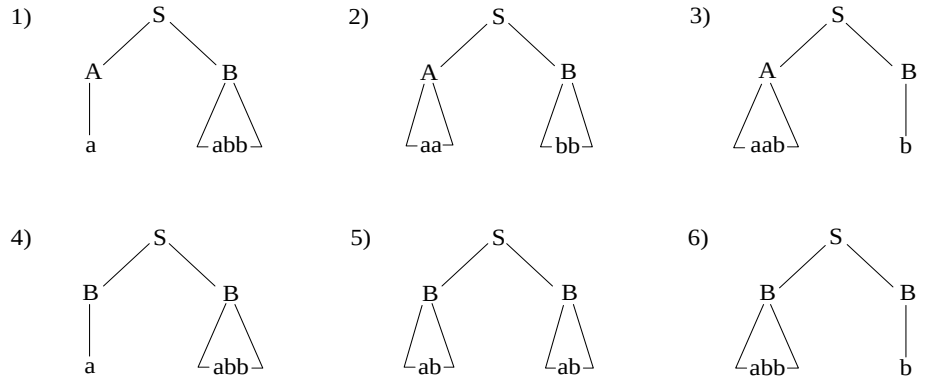
Tõestuseta (vt. A. Aho, J. Ullman The Theory of Parsing, Translation and Compiling. Vol.1: Parsing (Prentice-Hall Inc., 1972), Section 4.2.2 - venekeelne tõlge: kirjastus "Mir", Moskva 1978).

3.7.2. Cocke - Kasami - Youngeri algoritm (CKY - algoritm)

CKY algoritm lahendab süntaksanalüüsi ülesande KV-grammatikate korral, mis on Chomsky normaalkujul. Sellisel juhul analüüsitava sõna süntaksipuu on kahendpuu. Näiteks grammatika

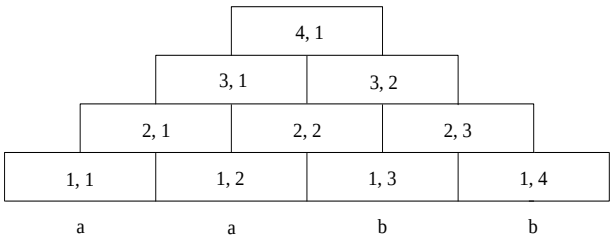
$$\begin{array}{lll} S \longrightarrow AB & S \longrightarrow BB & \\ A \longrightarrow CC & A \longrightarrow AB & A \longrightarrow a \\ B \longrightarrow BB & B \longrightarrow CA & B \longrightarrow b \\ C \longrightarrow BA & C \longrightarrow AA & C \longrightarrow b \end{array}$$

Tuletuspuu juure korral võib olla kasutatud üht kahest produktsioonist: $S \longrightarrow AB$ või $S \longrightarrow BB$. Kummastki juure vahetust alluvast võib tuletada analüüsitava sõna $w = x_1 \dots x_n$ m-täheline prefiksi ja (n - m)-tähelise sufiksi. Näiteks $w = aabb$ korral on antud grammatikas võimalik tuletuspuude konstrueerimisel lähtuda kuuest "lähendist":



Kõik kuus varianti ei pruugi realiseeruda, näiteks juhtum 2) korral pole võimalik ehitada tuletust $A \Longrightarrow^* aa$ ning seega ei leidu vasakut alampuud.

CKY algoritm "ühendab"erinevad võimalikud tuletuspuud, mis on ühe või teise analüüsitava stringi alamsõna süntaksipuudeks. Täpsemalt, algoritm täidab püramiidikujulise tabeli, mille alus koosneb n elemendist, selle kohal on $n-1$ elementi jne. Käesolevas punktis kasutatava grammatika näite ja analüüsitava sõna "$aabb$" korral on tabel järgmise kujuga (arvude paar tabeli elemendis esitab elemendi "aadresse"):



Tabeli lahtrisse aadressiga (i, j) paigutatakse mitteterminaal A parajasti siis, kui

$$A \Longrightarrow^* x_j x_{j+1} \dots x_{j+1-1} \tag{14}$$

Teiste sõnadega, A on elemendis aadressiga (i, j), kui temale alluva püramiidi alus "toetub" sõnale $x_j x_{j+1} \dots x_{j+1-1}$.

Kuna eeldame, et vaadeldav KV-grammatika on Chomsky normaalkujul, siis võib tuletus (14) olla vaid kujul:

$$A \Longrightarrow BC \Longrightarrow^* x_j x_{j+1} \dots x_{j+m-1} x_{j+m} \dots x_{j+i-1},$$

kus $m < i$ ja alamsõna $x_j \dots x_{j+m-1}$ on tuletatud mitteterminaalist B. Seega siis esineb sümbol A tabeli elemendis (i, j), kui mingi $m < i$ korral sisaldavad elemendid (m, j) ja $(i - m, j + m)$ vastavalt sümbolid B ja C ning grammatikas on produktsioon $A \longrightarrow BC$. Selline omadus ei kehti ainult tabeli esimese rea $((1, 1), (1, 2), \dots, (1, n))$ korral, kus sümbol A esineb elemendis $(1, j)$ parajasti siis, kui grammatika sisaldab produktsiooni $A \longrightarrow x_j$.

Tingimusest (14) järeldub vahetult, et $w \in \mathcal{L}(G)$ parajasti siis, kui sümbol S kuulub elemendi $(n, 1)$ koosseisu.

Esitatud arutluse võime kokkuvõtlikult esitada järgmise algoritmina.

Algoritm 10.

Antud:

- KV-grammatika $G = (\Sigma, N, P, S)$ Chomsky normaalkujul;
- string $w = x_1 \dots x_n \in \Sigma^*$.

Väljund: Mitteterminaalide hulkade maatriks $\| E_{ij} \|$, kus

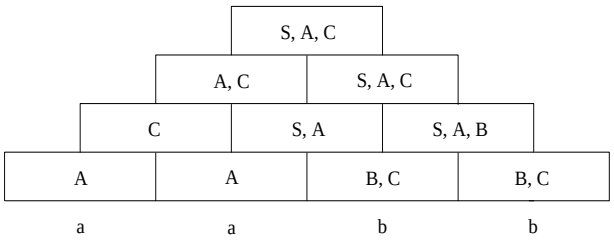
$$0 < i \leq n, \;\; 0 < j \leq n \;\; \text{ja} i + j \leq n + 1.$$

Meetod:

for $j := 1$ **to** n **do**
  if $A \longrightarrow x_j \in P$ **then**
    $E(1, j) := W(1, j) \cup \{A\}$;
for every $i < 1$ **do**
  for $j := 1$ **to** $n - 1 + 1$ **do**
    for $m := 1$ **to** $i - 1$ **do**
      if $E(m, j) = B \& E(i - m,\ j + m) = C \& A \longrightarrow BC \in P$
        then $E(i, j) = E(i, j) \cup \{A\}$.

□

Antud punkti algul toodud näite korral oleks tabeli E kuju:



Seega kuulub sõna "*aabb*" vaadeldavasse keelde.

Induktsiooniga tabeli reaindeksi järgi ning tuginedes ülalesitatud arutlusele, saab tõestada, et algoritm 10 on korrektne.

Teoreem 17. *Algoritmi 10 tulemusena $A \in E(i, j)$ parajasti siis, kui grammatikas G on esitatav tuletus $A \longrightarrow x_j x_{j+1}$. $w \in \mathcal{L}(G)$ parajasti siis, kui $S \in E(n, 1)$.*

Teoreem 18. *CKY-algoritmi ajaline keerukus on $\mathcal{O}(n^3)$, kus n on analüüsitava sõna pikkus.*

3.7.3. Magasinmäluga automaadid (pinuautomaadid)

Magasinmäluga automaat on seadeldis KV-keelte aktsepteerimiseks. Automaadil on sisendlint ja mälu. Sisendlindile kirjutatud sümboleid saab automaat ühekaupa lugeda e. skaneerida. Iga sümbolit loetakse üks kord ning lugemine toimub alates aktsepteeritava sõna algusest vasakult paremale. Mälu on organiseeritud magasinina, kuhu salvestatakse automaadi töö käigus läbitud olekud. Mälust lugemisel saadakse tulemusena viimati salvestatud oleku tähis. Töö algul on automaadi magasin tühi, s.t. magasinis on tühja magasini tunnusena tema põhja tähistav sümbol '$'. Masina tööd juhtigu programm, mis näitab, kuidas muuta mälu (magasini) seisu lindilt järjekordse sümboli lugemisel koos samaaegse sümboli lugemisega magasinist.

Näide 31. Olgu magasinimäluga automaadi käitumine määratud järgmiste korraldustega kujul

$$< a, A, X >,$$

kus a on lindilt loetav sümbol, A - magasinist loetav sümbol ja X on string, mis kirjutatakse tulemusena magasini. Olgu vaadeldava automaadi M_1 programmis seitse korraldust:

1. $< a, \$, A\$ >$
2. $< b, \$, B\$ >$
3. $< a, A, AA >$
4. $< b, B, BB >$
5. $< a, B, \epsilon >$
6. $< B, A, \epsilon >$
7. $< \epsilon, \$, \epsilon >$

Sisendlindilt skaneeritav sõna w nimetatakse aktsepteeritavaks (kuuluvaks keelde $(T(M_1))$, kui pärast viimase sümboli skaneerimist magasin on tühi. Antud automaat aktsepteerib sõnu hulgast

$$T(M_1) = \{w \mid w \in (a+b)^*, \text{ } w \text{ sisaldab võrdse arvu } a\text{-sid ja } b\text{-sid}\}.$$

(Kontrollige viimast väidet!)

□

Esitame nüüd magasinmäluga automaadi formaalse definitsiooni.

Definitsioon 44. Magasinmäluga automaat (MMA) *on struktuur*

$$M = (\Sigma, \Gamma, Q, p, q_0, \$, F),$$

kus

1. Σ - *sisendtähestik;*
2. Γ - *magasini tähestik;*
3. Q - *olekute tähestik;*
4. $p : (\Sigma \cup \{\epsilon\} \times \Gamma \times Q \longrightarrow \Gamma^* \times 2^Q$ - *üleminekufunktsioon;*
5. $q_0 \in Q$ - *lähteolek;*
6. $\$ \in \Gamma$ - *magasini lähtesümbol;*
7. $F \subseteq Q$ - *lõppolekute hulk.*

MMA $M = (\Sigma, \Gamma, Q, p, q_0, \$, F)$ *konfiguratsiooniks nimetatakse kolmikut*

$$(w, \gamma, q) \in \Sigma^* \times \Gamma^* \times Q,$$

kus sõna w *esitab sisendlindil veel skaneerimata alamstringi, magasini mälus salvestatud stringi ja* q *- automaadi jooksvat olekut. MMA* M *lähtekonfiguratsiooniks nimetame kolmikut* $(w, \$, q_0)$*; lõppkonfiguratsiooniks olgu* (ϵ, ϵ, r)*, kus* $r \in F$*.*

Definitsioon 45. *MMA* töötaktiks *nimetatakse binaarset seost* $\vdash$ *konfiguratsioonide hulgal, nii et*

$$(aw, Z\alpha, q) \vdash (w, \gamma\alpha, q'), \text{ kui } (\gamma, q') \in p\,(a, Z, q)$$

või

$$(w, Z\alpha, q) \vdash (w, \gamma\alpha, q'), \text{ kui } (\gamma, q') \in p\,(\epsilon, Z, q).$$

Definitsioon 46. *MMA* $M = (\Sigma, \Gamma, Q, p, q_0, \$, F)$ *poolt aktsepteeritav keel on hulk*

$$T(M) = \{w \mid (w, \$, q_0) \vdash^* (\epsilon, \epsilon, r),\ r \in F\}.$$

Näide 32. KV-keele $L = \{0^n 1^n \mid n \geq 0\}$ aktsepteerimiseks sobib

$$M\ M\ A\ M_2 = (\Sigma, \Gamma, Q, p, q_0, \$, F),$$

kus

$$\Sigma = \{0, 1\},\ \Gamma = \{0, 1\},\ Q = \{q_0, q_1, q_2\},\ F = \{q_0\}$$

ja üleminekufunktsioon on määratud seostega:

$$\begin{array}{ll} p\,(0, \$, q_0) = \{(0\$, q_1)\} & p(1, 0, q_2) = \{(q_2, \epsilon)\} \\ p\,(0, 0, q_1) = \{(00, q_1)\} & p\,(\epsilon, \$, q_2) = \{(q_0, \epsilon)\} \\ p\,(1, 0, q_1) = \{(\epsilon, q_2)\} & \end{array}$$

□

Teoreem 19. *Iga KV-keele* L *jaoks leidub MMA* M*, nii et* $L = T(M)$*.*

Tõestus. Olgu $L = \mathcal{L}(G)$, kus $G = (\Sigma, N, P, S)$ on KV-grammatika. Konstrueerime automaadi $M = (\Sigma, \Gamma, Q, p, q_0, \$, F)$ järgmiselt:

1. $\Gamma := \Sigma \cup, N \cup, \{\$\}$ (eeldame, et $\$ \notin N \cup \Sigma$);
2. $Q = \{q_0, q_1, q_2\}$;
3. Määrame üleminekufunktsiooni nii, et
   $p\,(\epsilon, \$, q_0) = \{(S\$, q_1)\}$;
   $p\,(\epsilon, A, q) = \{(w, q_1) \mid A \longrightarrow w \in P,\ A \in N\}$;
   $p\,(a, a, q_1) = \{(\epsilon, q_1)\}$, kui $a \in \Sigma$;
   $p\,(\epsilon, \$, q_1) = \{(\epsilon, q_2)\}$;
4. $F := \{q_2\}$.

vaatleme sõna vasakderivatsiooni grammatikas G:

$$S \Longrightarrow v_1 A_1 \alpha_1 \Longrightarrow v_1 v_2 A_2 \alpha_2 \alpha_1 \Longrightarrow \ldots \Longrightarrow v_1 v_2 \ldots v_n = x$$

Vastavalt esitatud konstruktsioonile on masina M töö sõna x aktsepteerimisel vaadeldav järgmise taktide jadana:

$$\begin{array}{l} (v_1 v_2 \ldots v_n, q_0) \vdash (v_1 v_2 \ldots v_n, S\$, q_1) \vdash (v_1 v_2 \ldots v_n, v_1 A_1 \alpha_1 \$, q_1) \vdash^* \\ \quad \vdash^* (v_2 \ldots v_n, A_1 \alpha_1 \$, q_1) \vdash (v_2 \ldots v_n, v_2 A_2 \alpha_2 \alpha_1 \$, q_1) \vdash^* \\ \quad \vdash^* (v_3 \ldots v_n, v_3 A_2 \alpha_2 \alpha_1 \$, q_1) \vdash \ldots \\ \qquad \ldots \\ \qquad \ldots \vdash (v_n, v_n \$, q_1) \vdash^* (\epsilon, \$, q_1) \vdash (\epsilon, \epsilon, q_2). \end{array}$$

Seega $x \in \mathcal{L}(G)$ parajasti siis, kui $(x, \$, q_0) \vdash^* (\epsilon, \epsilon, q_2)$ e. $x \in T(M)$.

□

Näide 33. Konstrueerime MMA KV-grammatikaga

$$\begin{array}{l} S \longrightarrow aSb \\ S \longrightarrow \epsilon \end{array}$$

genereeritava keele aktsepteerimiseks. Vastava MMA üleminekufunktsioon p rahuldab tingimusi:

$$\begin{array}{ll} p\,(\epsilon, \$, q_0) = \{(0\$, q_1)\} & p(b, b, q_1) = \{(\epsilon, q_1)\} \\ p\,(\epsilon, S, q_1) = \{(aSb, q_1)\},\ (\epsilon, q_1)\} & p\,(\epsilon, \$, q_1) = \{(\epsilon, q_2)\} \\ p\,(a, a, q_1) = \{(\epsilon, q_1)\} & \end{array}$$

□

Definitsioon 47. *MMA M' ja M'' on ekvivalentsed, kui $T(M') = T(M'')$.*

Lemma 9. *Iga MMA M jaoks leidub ühe olekuga MMA M', nii et $T(M) = T(M')$.*

Tõestus. Eeldame, et MMA $M = (\Sigma, \Gamma, Q, p, q_0, \$, F)$ on üks lõppolek. (Iga automaati saab selliseks teisendada. Kuidas?). Konstrueerime uue MMA

$$M' = (\sigma, \Gamma', Q', p', *, \#, \{*\}),$$

millel on üks olek $*$ (s.t. $Q' = \{*\}$).

Automaadi M' magasini tähestiku moodustagu sümbolid $[sAt]$, mida interpreteeritakse järgmiselt:

magasinist sümboli A lugemisel võib automaat M siirduda olekust s olekusse t.

Teiste sõnadega, $[sAt]$ tähistab asjaolu, et mingi terminaalse sümboli $a \in \Sigma$ korral $(\gamma, t) \in (a, A, s)$, kus $\gamma \in \Gamma'$ ning $s, t \in Q$.

Uue automaadi üleminekufunktsioon olgu määratud nii, et oleks täidetud järgmised seosed 1 - 3 ja ainult need:

1. $([tBx_1][x_1Cx_2][x_2Dx_3], *) \in p'(a, [sAx_3], *) \quad \forall x_1, x_2, x_3 \in Q$
   kui automaadi M korral $(BCD, t) \in p\,(a, A, s)$;
2. $(\epsilon, *) \in p'(a, [sAt], *)$,
   kui automaadi M korral kehtib seos $(\epsilon, t) \in p\,(a, A, s)$;
3. $p'(\epsilon, \#, *) = \{([q_0\$x], *) \mid x \in Q\}$.

Iga lähtesõna w korral algab masina M' töö $*$ sammudega

$$(w, \#, *) \vdash (w, [q_0, \$q], *) \vdash \ldots$$

Seame igale masina M' konfiguratsioonile

$$(w, [q_1A_1q_2][q_2A_2q_3] \ldots [q_kA_kq_{q+1}], *)$$

vastavusse korteeži

$$< w, A_1, A_2 \ldots A_k, q_1 > .$$

Erijuhul $(w, \#, *)$ olgu vastav korteež kujul $< w, \#, - >$.

Võrduse $T(M) = T(M')$ tõestamiseks piisab näidata, et automaadi M korral kehtib seos

$$(w, \$, q_0) \vdash^* (w', \gamma, q)$$

parajasti siis, kui arvestades ülaltoodud tähistusi, kehtib automaadi M' korral

$$< w, \#, - > \vdash^* \ (w', \gamma, q). \tag{15}$$

Siis kehtib erijuhul $w \in T(M)$ e. $(w, \$, q_0) \vdash^* (\epsilon, \epsilon, r)$, $r \in F$ parajasti siis, kui automaadi M' jaoks $< w, \#, - > \vdash^* < \epsilon, \epsilon, r >$ e. $w \in T(M')$.

Vastava tõestuse esitame vaid sisalduvuse $T(M) \subseteq (M')$ juhu jaoks. Sisalduvus $T(M') \subseteq t(M)$ tõestatakse analoogiliselt.

Olgu seoses (15) taktide arv $k = 0$. Seega (15) omandab kuju

$$(w, \$.q_0) \vdash^0 (w, \$, q_0),$$

millele automaadi M' korral vastab tuletus

$$(w, \#, *) \vdash (w, [q_0\$q], *)$$

e. kasutatavate tähistuste raames:

$$< w, \#, - > \vdash < w, \$, q_0 > .$$

Järelikult on induktsiooni baas tõestatud.

Eeldame nüüd, et tõestamist vajav väide kehtib, kui automaadi M korral on tehtud i takti. Vaatleme nüüd $i + 1$ taktilist jada:
Juht A:

$$(w, \$, q_0) \vdash^i (aw', A\gamma, q_1) \vdash (w', BCD\gamma, q_{i+1}).$$

Selline jada on võimalik vaid juhul, kui

$$(BCD, q_{i+1}) \in p\ (a, A, q_1).$$

Vastavalt masina M' konstruktsioonile kehtib siis ka seos

$$([q_{i+1}Bp_1]\ [p_2Cp_3]\ [p_3Dp_4], *) \in p'(a, [q_1Ap_4], *)$$

mingite $p_1, p_2, p_3, p_4 \in Q$ korral.

Arvestades induktsiooni eeldust ja viimatileitud tingimust, saame, et masina M' korral kehtib seos

$$< w, \#, - > \vdash^* < aw', A\gamma, q_i > \vdash < w', BCD\gamma, q_{i+1} > .$$

Juht B: Automaadi M töö $i + 1$ takti jooksul on kirjeldatav jadaga

$$(w, \$, q_0) \vdash (aw', A\gamma, q_i) \vdash (w', \gamma, q_{i+1}).$$

Sellisel juhul on vastavalt automaadi M' konstruktsiooni p. 2 saadav jada

$$< w, \#, - > \vdash^* < aw', A\gamma, q_1 > \vdash < w', \gamma, q_{i+1} > .$$

□

Ühe olekuga MMA üleminekufunktsioonis ja konfiguratsioonides võib oleku tähise $*$ ka ära jätta, kirjutades

$$p'(a, A, *) = (BCD, *)$$

asemel

$$p'(a, A) = BCD.$$

Näide 34. Konstrueerime ekvivalentse ühe olekuga automaadi MMA-le

$$M = (\{a, b\}, \{a, b, \$\}, \{q, s, r\}, p, q, \$, \{r\}),$$

kus üleminekufunktsioon on esitatud seostega:

$$\begin{array}{ll}
p\ (a, \$, q) = \{(a\$, q)\} & p(b, b, q) = \{(bb, q), (\epsilon, s)\} \\
p\ (b, \$, q) = \{(b\$, q)\} & p\ (a, a, s) = \{(\epsilon, s)\} \\
p\ (a, a, q) = \{(aa, q), (\epsilon, s)\} & p\ (b, b, s) = \{(\epsilon, s)\} \\
p\ (b, a, q) = \{(ba, q)\} & p\ (\epsilon, \$, s) = \{(\epsilon, r)\} \\
p\ (a, b, q) = \{(ab, q)\} &
\end{array}$$

(Toodud automaat aktsepteerib keelt $L = \{xx^r \mid x \in \{a, b\}^*\}$.)

Konstrueeritava ühe olekuga automaadi magasini sümbolite lühemaks esitamiseks toome sisse uued tähised:

$$Z = [q\$r] \qquad E = [qbs]$$
$$A = [q\$q] \qquad F = [sas]$$
$$B = [qaq] \qquad G = [sbs]$$
$$C = [qas] \qquad H = [s\$r]$$
$$D = [qbq]$$

Vastavalt lemma 9 tõestuses toodud konstruktsioonile saame ülemineku

$$p\,(a, \$, q) = \{(a\$, q)\}$$

jaoks uue üleminekufunktsiooni:

$p'(a, [q\$q]) = \underline{\{([qaq]\ [q\$q], *)}, \ ([qas]\ [s\$q], *), \ ([qar]\ [r\$q], *)\}$
$p'(a, [q\$s]) = \{([qaq]\ [q\$s], *), \ ([qas]\ [s\$s], *), \ ([qar]\ [r\$s], *)\}$
$p'(a, [q\$r]) = \underline{\{([qaq]\ ]q\$r], *)}, \ \underline{([qas]\ [s\$r], *)}, \ ([qar]\ [r\$r], *)\}$

Pidades silmas sümboli $[sAt]$ interpretatsiooni ("Kui magasini tipus on sümbol A, siis on võimalik üleminek olekust s olekusse t"), on eelnevas ilmselt sümboleid, milliseid kunagi automaadi M' magasini ei kirjutata, sest lähteautomaadis vastavat üleminekut kunagi ei toimu. Näiteks ei saa lähteautomaat kunagi pöörduda olekust s tagasi olekusse q. Seega on näiteks mõttetu sümbol $[s\$q]$. Täpsem analüüs näitab, et eelnevas on "kasulikud" üleminekud vaid need, mis on allajoonitud.

Kasutades automaadi M' jaoks sissetoodud tähistusi, saame kokkuvõttes, et ülemineku $p\,(a, \$, q) = \{(a\$, q)\}$ vastavad automaadis M' üleminekud

$$p'(a, A) = \{BA\}$$

ja

$$p'(a, Z) = \{BZ, \ \ CH\}.$$

Viies läbi analoogilise konstruktsiooni kõigi automaadi M üleminekute jaoks, saame tulemuseks kõik automaadi M' üleminekud, mis kokkuvõtlikult on esitatavad tabelina:

Lindi sümbol	Magasini tipp								
	A	B	C	D	E	F	G	H	Z
a	BA	BB	BC, CF, ϵ	BD	BE, CG	ϵ	-	-	BZ, CH
b	DA	DB	DC, EF	DD	DE, EG, ϵ	-	ϵ	-	DZ, EH
ϵ	-	-	-	-	-	-	-	ϵ	-

Eeldusel, et saadud automaadi magasin on töö lõppedes tühi, võib automaadi veelgi minimeerida: kui magasini tipus on A, B või D, pole saavutatav olukord, kus magasin saab tühjaks. Seega oleks uus, ühe olekuga MMA lõplikul kujul:

Sisend-sümbol	Magasini tipp					
	C	E	F	G	H	Z
a	CF, ϵ	CG	ϵ	-	-	CH
b	EF	EG, ϵ	-	ϵ	-	EH
ϵ	-	-	-	-	ϵ	-

Näiteks sisendstringi $'aabaabaa'$ korral oleks saadud automaadi töö esitatav järgmise konfiguratsioonide jadaga:

$(aabaabaa,\ Z)$
$(abaabaa,\ CH)$
$(baabaa,\ CFH)$
$(aabaa,\ EFFH)$
$(abaa,\ CGFFH)$
$(baa,\ GFFH)$
$(aa,\ FFH)$
$(a,\ FH)$
$(\epsilon,\ F)$
$(\epsilon,\ \epsilon)$

□

Teoreem 20. *Iga MMA M jaoks leidub KV-grammatika G, nii et $T(M) = \mathcal{L}(G)$.*

Tõestus. Vastavalt lemmale 9 võime eeldada, et MMA M on ühe olekuga: $M = (\Sigma, \Gamma, \{*\}, p, *, Z, \{*\})$.

Konstrueerime KV-grammatika $G = (\Sigma, N, P, S)$, võttes $N = \Gamma$, $S = Z$ ning määrates produktsioonide hulga P vastavalt alljärgnevatele tingimutele 1 ja 2:

1. $A \longrightarrow aB_1 \dots B_k \in P$, kui $B_1 \dots B_k \in p\,(a, A)$, $k \geq 0$;
2. $A \longrightarrow B_1 \dots B_k \in P$, kui $B_1 \dots B_k \in p\,(\epsilon, A)$, $k \geq 0$.

Kasutades matemaatilist induktsiooni, on lihtne näidata, et automaadi M korral kehtib $(x, A) \vdash^* (\epsilon, \epsilon)$ parajasti siis, kui $A \Longrightarrow_G^* x$. Jätame selle lugejale harjutusülesandeks.

□

Näide 35. Näites 34 esitatud MMA-le vastab KV-grammatika

$$\begin{array}{ll} S \longrightarrow aCH & E \longrightarrow b \\ S \longrightarrow bEH & E \longrightarrow bEG \\ C \longrightarrow a & F \longrightarrow a \\ C \longrightarrow aCF & G \longrightarrow b \\ C \longrightarrow bEF & H \longrightarrow \epsilon \\ E \longrightarrow aCG & \end{array}$$

Pärast teisendamist Greibachi normaalkujule sisaldab see grammatika produktsioonid

$$\begin{array}{ll} S \longrightarrow aC & S \longrightarrow bE \\ C \longrightarrow aC & C \longrightarrow bEF \\ C \longrightarrow a & E \longrightarrow bEG \\ F \longrightarrow aCG & E \longrightarrow b \\ F \longrightarrow a & G \longrightarrow b \end{array}$$

□

3.8. KV-keelte tarvilikkuse tingimus

Süntaksipuu kõrguseks nimetatakse temas leiduva pikima tee kaarte arvu.

Lemma 10. *Tähistame KV-grammatikas $G = (\Sigma, N, P, S)$ sümboliga*

$$m = \max_{A \longrightarrow w \in P} \mid w \mid .$$

Olgu sõna x tuletuspuu kõrgus j. Siis $\mid x \mid \leq m^j$.

Tõestus. Näitame, et suvalise süntaksipuu T korral $\mid \mathrm{Kr}(T) \mid \leq m^j$. Kasutame selleks matemaatilist induktsiooni.

$j = 1$ korral on tegu elementaarpuuga, mille korral tõestatav väide kehtib triviaalselt.

Oletame, et tõestatav väide kehtib puude korral, mille kõrgus on kuni $k-1$. Olgu sõna w süntaksipuu kõrgus k. Selle puu juurel on maksimaalselt m vahetut alluvat, millele omakorda alluvate alampuude kõrgus on maksimaalselt $k-1$. Seega on nende kroonide pikkus kuni m^{k-1}. Kogu puu T krooni jaoks kehtib aga võrratus:

$$\mid \mathrm{Kr}(T) \mid \leq m \cdot m^{k-1} = m^k.$$

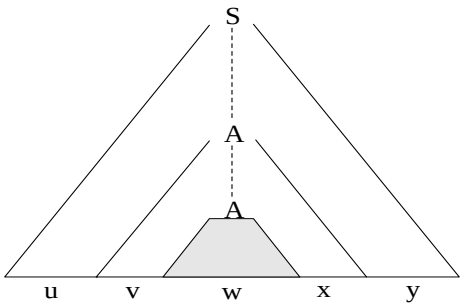
Teoreem 21. *(KV-keelte tarvilikkuse tingimus).*

Olgu L KV-keel. Siis leiduvad ainult keelest L sõltuvad konstandid p ja q, nii et $z \in L$ ja $\mid z \mid > p$ korral leidub jaotus $z = uvwxy$, kusjuures

1. $\mid vwx \mid \leq q$;
2. *v ja x pole korraga tühjad sõnad;*
3. *iga $i \geq 0$ korral $uv^iwx^iy \in L$.*

Tõestus. Olgu $G = (\Sigma, N, P, S)$ selline KV-grammatika, et $L = \mathcal{L}(G)$, kusjuures m tähistagu maksimaalset produktsiooni parema poole pikkust. Olgu $k = \mid N \mid$.

Valime arvuks $p = m^k$. Sõna $z \in \mathcal{L}(G)$, mille korral $\mid z \mid > p$. Tuletuspuu kõrgus peab vastavalt lemmale 10 olema suurem kui $k+1$. Vaadeldes sellise tuletuspuu pikimat teed, näeme, et mingi mitteterminaal, näiteks A, peab esinema sellel teel vähemalt kaks korda. Valime kaks "alumist" sümboli A esinemist ning jaotame sõna z alamsõnadeks $u\ v\ w\ x\ y$, nagu näidatud joonisel.



Sõnale vwx vastava alampuu kõrgus on väiksem kui $k+1$. Seega võime valida konstandi $q = m^{k+1} \geq \mid vwx \mid$. Ilmselt võib esitatud tuletuspuust eemaldada sõnale vwx vastava alampuu ning asendada ta sõna w alampuuga. Saadud tuletuspuu kuulub grammatikasse G ning järelikult on talle vastav terminaalne sõna $uwy = uv^0wx^0y$ tuletatav grammatikas G.

Asendades aga joonisel viirutatud alampuu sõna vwx alampuuga, näeme, et ka sõnad uv^2wx^2y, uv^3wx^3y jne. kuuluvad keelde L.

Lõpuks märgime veel, et ei saa kehtida võrdused $v = x = \epsilon$. Kui selline seos kehtiks, saaksime sõna asendada vwx süntaksipuu viirutatud alampuuga, ilma et tuletatav sõna muutuks ($uwy = uvwxy$). Seega saaksime sellele sõnale tuletuspuu, mille kõrgus on väiksem kui $k+1$. Saadud tulemus on aga vastuolus eeldusega $\mid z \mid > p$.

□

Näide 36. Keel $L = \{a^nb^nc^n \mid n > 0\}$ pole KV-keel.

Tõepoolest, kui oletada, et L on KV-keel, siis peavad leiduma konstandid p ja q, sellised, et nad rahuldaksid teoreemi 21 tingimusi.

Olgu $r > p$, $r > q$ ning püüame leida sõna $a^r b^r c^r$ jaotust $a^r b^r c^r = uvwxy$.

Kui sõna vx ei sisalda üht tähtedest, näiteks c, on sõnas uv^2wx^2y c-de arv väiksem kui a-de või b-de arv. Kui aga sõna vx sisaldab kõiki kolme, peab ühes neist olema kaks erinevat tähte. oletades, et v sisaldab a-d ja b-d, saame jälle vastuolu, sest sõnas uv^2wx^2y ei ole kõik a-d enam enne b-sid.

□

Näide 37. Keel $L = \{a^{n^2} \mid n > 0\}$ pole KV-keel.

Oletame vastuväiteliselt, et tegu on KV-keelega. Valime $p \geq q$ ning vaatleme sõna $Z = a^{p^2} \in L$. Et $\mid z \mid > p$, siis peab leiduma jaotus $z = vwuxy$, nii et $\mid vwx \mid < q \leq p$.

Koostame võrratuse:

$$p^2 < \mid uv^2wx^2y \mid \leq \mid uv^2w^2x^2y \mid = \mid z \mid + \mid vwx \mid < p^2 + q < p^2 + 2p + 1 = (p+1)^2.$$

Järelikult

$$p^2 < \mid uv^2wx^2y \mid < (p+1)^2.$$

Seega ei saa sõna uv^2wx^2y kuuluda keelde L mistahes konstantide $p \geq q$ korral ning järelikult pole keel L KV-keel.

□

Järeldus 2. *Sisalduvus $\mathcal{L}_2 \subset \mathcal{L}_1$ on range.*

3.9. KV-keelte semantika

Süntaksanalüüs moodustab KV-keelte transleerimisprotsessi

$$Tr : L_1 \longrightarrow L_2$$

esimese osa. Lähtesõna x süntaktilise õigsuse kontroll ning tuletuspuu koostamine moodustavad kujutuse Tr realisatsioonist väiksema osa (ca 20 - 30 % kogu töömahust), suurem osa töömahust läheb sünteesile (semantiline analüüs ja teksti genereerimine - vt. transleerimisprotsessi faasid, p. 3.6). Transleerimise sünteesi osa nimetatakse ka *semantiliseks töötluseks.*

Lisaks suuremale töömahukusele on semantilise töötluse formaalse spetsifitseerimise ja realiseerimise meetodid vähem läbi uuritud. Sõltuvalt töötluse eesmärkidest kasutatakse programmide semantika defineerimiseks kolme erinevat lähenemist: operatsiooniline, denotatsiooniline ja aksiomaatiline semantika. Nimetatud tehnikate olemus ning võrdlus on kursuste "Programmide semantika" ja "Teoreetiline informaatika II" aineks (TTÜ ainesüsteemis ained koodidega vastavalt LAI 5151 ja LAI 5160). Käesolevas kursuses vaatleme programmeerimiskeelte semantika realiseerimisel enamkasutatavat atribuuttehnikat.

3.9.1. Atribuutgrammatika mõiste

Üldjuhul esitab lause mingi seose teatud objektide vahel. Programmeerimiskeelte korral on neiks objektideks arvutis salvestatavad ning töödeldavad andmeobjektid: arvud, massiivid, järjekorrad, failid, programmid jne. Meditsiinilises tekstis võivad olla kujutatud inimese kehaosad ja organid, haigused ning nende tunnused, ravimid jne. Teisisõnu, igal keelel on nn. probleem-valdkond, mille objektidevahelisi suhteid antud keeles saab väljendda. Sellist objektide kogumit nimetame edaspidi keele *semantiliseks piirkonnaks.*

Keele L lause semantika e. tähenduse spetsifitseerimiseks tuleb kirjeldada vastavus keele lausete (KV-keelte korral lausete tuletuspuude) ja semantilise piirkonna D objektide vahel:

$$\varphi : L \longrightarrow D.$$

semantilise piirkonna D kirjeldamiseks kasutame nn. semantilisi muutujaid hulgast

$$X = \{x_1, x_2, \ldots\},$$

millest igaühe jaoks on defineeritud muutumispiirkond (doomen) hulgas D. Seega on semantilise muutuja x_i $(i = 1, 2, \ldots)$ jaoks määratud võimalike väärtuste hulk

$$D(x_i) = Dx_i \subseteq D.$$

Antud käsitluses eeldatakse, et iga semantilise muutuja väärtuste hulk on osaliselt järjestatud ning sisaldab nn. "täielikult määramata väärtuse" **nil** (Kui muutuja x jooksvaks väärtuseks on **nil** $\in D_x$, tähendab see, et seni pole muutuja x väärtust veel arvutatud). Kujutuse φ määratlemiseks kasutatakse atribuuttehnikas formalismi, mida nimetatakse atribuutgrammatikaks.

Definitsioon 48. Atribuutgrammatika *on struktuur* $AG = (G, A, R)$, *kus*

- $G = (\Sigma, N, P, S_0)$ *on KV-grammatika;*
- $V \longrightarrow P(X)$ *on KV-grammatika* G *sümbolite hulga* $V = \Sigma \cup N$ atribuutide *hulk;*
- $R = \{R_p\}_{p \in P}$ *on semantikareeglite pere. Iga KV-grammatika* G *produktsiooni semantikareegel* R_p *on omistamiste hulk, mis määrab, kuidas arvutada tuletuspuu atribuute (vt. definitsioon 49 allpool).*

Lihtsuse mõttes eeldame, et sümboli $Y \in \Sigma \cup N$ atribuutide hulk $A(Y)$ on lõplik, s.t.

$$A(Y) = \{a_1, a_2, \ldots, a_{n_Y}\},$$

kus $a_1, a_2, \ldots, a_{n_Y}$ on semantilised muutujad hulgast X.

Käsitluse lihtsustamise huvides eeldatakse veel, et iga sümboli $Y \in \Sigma \cup N$ atribuutide hulk on jaotatud kaheks:

$$A(Y) = I(Y) \cup S(Y),$$

nii et $I(Y) \cap S(Y) = \emptyset$. Muutujaid hulgast $I(Y)$ nimetatakse päritud atribuutideks ja muutujaid hulgast $S(Y)$ sünteesitud atribuutideks.

Põhimõtteliselt defineerib kujutus A atribuudid ka igale süntaksipuule (sealhulgas ka grammatika produktsioonile, kuna see on vaadeldav elementaarse süntaksipuuna). Süntaksipuu t tipu n atribuutide hulk olgu $A(n) = A$, kui tipp n on märgendatud sümboliga $Y \in \Sigma \cup N$ ja $A(Y) = A$. Tipu n atribuuti a nimetatakse mõnikord ka atribuudi a esinemiseks puus T (ingl. k. - occurence of attribute; instance of attribute). Atribuudi a esinemist tipus n tähistatakse enamasti kui muutujat $Y.a$, kus Y on tipu n märgistus. Juhul kui puus on mitu sama märgendiga tippu, lisatakse atribuudi esinemise tähistuses märgendile indeks. Seega on atribuudi a erinevad esinemised (erinevate, kuid ühe ja sama sümboliga Y märgendatud tippude korral) $Y_0.a, Y_1.a, Y_2.a, \ldots$. Olgu KV-grammatika G produktsioon p esitatav kujul

$$X_0 \longrightarrow X_1, X_2, \ldots X_{n_p},$$

kus $X_0 \in N$ ja $X_i \in V \quad i > 0$ korral. Atribuutide hulgad olgu määratud järgmiselt:

$$\begin{aligned} A(X_0) &= \{a_1, a_2, \ldots, a_k\} \\ A(X_1) &= \{b_1, b_2, \ldots, b_e\} \\ \ldots\ldots\ldots\ldots &\ldots \ldots\ldots\ldots\ldots\ldots \\ A(X_{n_p}) &= \{d_1, d_2, \ldots, d_n\} \end{aligned}$$

Sel juhul koosneb produktsiooni p kõigi atribuutide (esinemiste) hulk järgmistest elementidest:

$X_0.a_1, X_0.a_2, \ldots, X_0.a_k,$
$X_1.b_1, X_1.b_2, \ldots, X_1.b_e,$
.....
$X_{n_p}.d_1, X_{n_p}.d_2, \ldots, X_{n_p}.d_n.$

Nende tähistuste korral saame esitada ka produktsiooni p semantikareegli formaalse definitsiooni:

Definitsioon 49. *Produktsiooni* $p : X_0 \longrightarrow X_1 \ldots X_{n_p}$ semantikareegel *on omistamiste hulk*

$$R_p = \{X_0.s := f_s(X_0.a_1, \ldots, X_{n_p}.d_n) \mid s \in S(X_0)\} \cup$$
$$\cup \{X_k.i := f_i(X_0.a_1, \ldots, X_{n_p}.d_n) \mid 0 < k \leq n_p, \quad i \in I(X_k)\},$$

kus f_s *ja* f_i *on antud semantilises piirkonnas määratud avaldised.*

Näide 38. Olgu $G = (\Sigma, N, P, F)$ KV-grammatika, mis genereerib kahendsüsteemi ratsionaalarvud[1]

$$\Sigma = \{0, 1, \cdot\}$$
$$N = \{F, L, B\}$$
$$P = \begin{cases} F \longrightarrow L.L & L \longrightarrow LB & B \longrightarrow 0 \\ F \longrightarrow L & L \longrightarrow B & B \longrightarrow 1 \end{cases}$$

Grammatika G terminaalidele pole atribuute lisatud. Mitteterminaalide atribuudid olgu määratud järgmiselt:[2]

$$S(F) = \{v : \mathbb{Q}\} \qquad I(F) = \emptyset$$
$$S(L) = \{v : \mathbb{Q}, l : \mathbb{N}\} \qquad I(L) = \{c : \mathbb{Z}\}$$
$$S(B) = \{v : \mathbb{Q}\} \qquad I(B) = \{c : \mathbb{Z}\}.$$

Semantikareeglid, mis määravad, kuidas arvutada grammatikaga G esitatud kahendarvude vääruse (tähenduse) kümnendsüsteemis, on toodud järgmises tabelis:

Produktsioon p	R_p
$F \longrightarrow L.L$	$F.v := L_1.v + L_2.v$ $L_1.c := 0$ $L_2.c := -L_2.l$
$F \longrightarrow L$	$F.v := L.v$ $L.c := 0$
$L \longrightarrow LB$	$L_0.v := L_1.v + B.v$ $L_0.l := L_1 l + 1$ $L_1.c := L_0.c + 1$ $B.c := L_0.c$
$L \longrightarrow B$	$L.v := B.v$ $L.l := 1$ $B.c := L.c$
$B \longrightarrow 0$	$B.v := 0$
$B \longrightarrow 1$	$B.v := 2^{B.c}$

[1] Näide on võetud artiklist D.E. Knuth. *Semantics of Context-Free Languages*, Mathematical Systems Theory, **2**, 2, 1968, pp. 127 - 146. Atribuuttehnikast huvitatuil on soovitav seda artiklit lugeda. Venekeene tõlge on ilmunud kogumikus M. Kurotškin (toim.). Semantika jazõkov programmirovania. "Mir", M. 1980.

[2] Avaldis $A = \{x_1 : D_1, \ldots, x_n : D_n\}$ tähistab siin asjaolu, et hulk A koosneb muutujatest $x_1, \ldots, x_n$, mille määramispiirkonnad on vastavalt $D_1, \ldots, D_n$. Iga D_i on semantiline väärtuste piirkond, s.t sisaldab elementi **nil**. Nii on käesoleval juhul $\mathbb{N} = \{\mathbf{nil}, 0, 1, 2, \ldots\}$. Analoogne laiendus on ka ratsionaalarvude ja täisarvude korral.

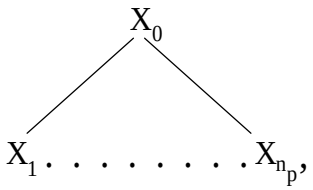
□

3.9.2. Süntaksipuu dekoreerimine

Atribuutgrammatika määrab reeglid, kuidas arvutada sõna $x \in \mathcal{L}(G)$ semantikat. Selleks tuleb arvutada sõna x tuletuspuu tippude atribuudid. Vastavat arvutusprotsessi nimetatakse *tuletuspuu dekoreerimiseks*. Lühidalt, dekoreerimine on protsess, mis seni, kuni saab arvutada uusi atribuutide väärtusi, täidab tsükliliselt tingimuslikku operaatorit

$$\textbf{if } (X_1.b = Z_b)\& \ldots \&(X_{n_p}.d = Z_d)\&(X_0.a = \textbf{nil})$$
$$X_0.a := f(Z_1, \ldots, Z_n).$$

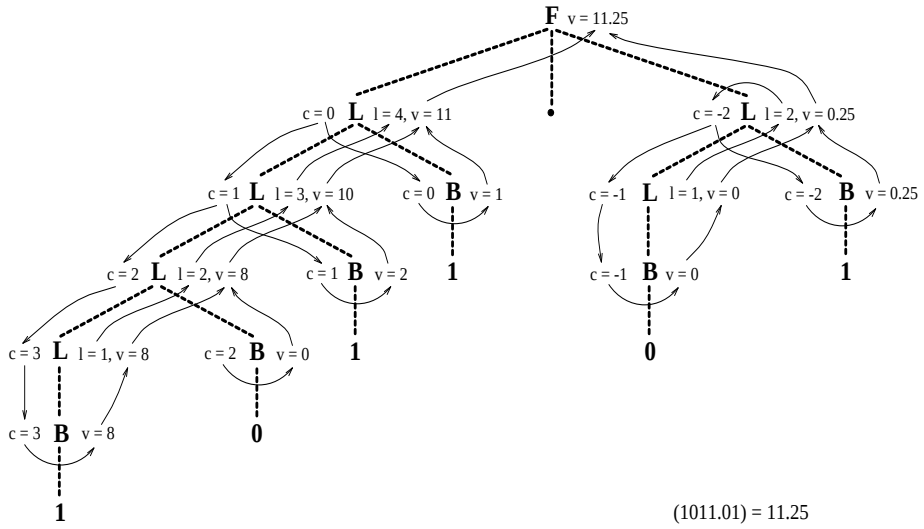
Siinjuure seeldatakse, et $Z_b \neq \textbf{nil}, \ldots, Z_d \neq \textbf{nil}$ ja $Z_b \in D_b, \ldots, Z_d \in D_d$ ning et süntaksipuus esineb elementaarpuu



mis vastab produktsioonile $p: \quad X_0 \longrightarrow X_1 \ldots X_{n_p}$. Semantikareegel R_p peaks sisaldama omistamise

$$X_0.a := f(X_1.b, \ldots, X_n.d).$$

Dekoreerimise lähteseisus olgu puu kõigi atribuutide väärtus **nil**. Joonisel 1 on esitatud sõna *1011.01* süntaksipuu dekoreerimise tulemus vastavalt näites 38 esitatud atribuutgrammatikale. Katkendjoontega on esitatud sõna süntaktiline struktuur, nooled esitavad atribuutide esinemiste väärtustamise järjekorda.



Joon. 1. Süntaksipuu dekoreerimine

Definitsioonis 49 esitatud tingimused garanteerivad, et iga tuletuspuu atribuudi arvutamiseks leidub täpselt üks omistamine. Samal ajal ei ole võimalik efektiivselt esitada kitsendust, mis garanteeriks, et atribuudid ei ole tsüklilises sõltuvuses ning iga tuletuspuu kõiki atribuute saab üldse arvutada (Joonisel 1 oleks näites 38 toodud grammatika ebaõige määratluse korral tekkinud atribuutide arvutamise järjekorda näitavatest nooltest tsükliline graaf).

Definitsioon 50. *Atribuutgrammatika* $AG = (G, A, R)$ *on korrektselt määratud, kui iga sõna* $x \in \mathcal{L}(G)$ *korral on dekoreeritud tuletuspuus kõigi atribuutide väärtused määratud (s.t.* $x \neq$ **nil***).*

Definitsioon 51. *Sõna* $x \in \mathcal{L}(G)$ semantika *on väärtus*

$$\varphi(X, K_1, \ldots, K_m) =< Z_1, \ldots, Z_n >,$$

kus $Z_1, \ldots, Z_n$ *on sõna* x *dekoreeritud tuletuspuu juure sünteesitud atribuutide väärtused ning* $K_1, \ldots, K_m$ *puu juure päritud atribuutide väärtused.*

Puu juure päritud atribuutide väärtused antakse dekoreerimisalgoritmile ette kui parameetrid, mis määravad sõna x konteksti. Seega esitab atribuutgrammatika semantika transleeritava sõna struktuuri ja konteksti põhjal. Näites 38 esitatud grammatika ei kasutanud lähtesõna välimist konteksti ning määrab kahendarvude semantika φ selliselt, et näiteks $\varphi(1011.01) = 11.25$.

4. Ülesannete lahenduvus

4.1. Turingi masin

Kontekstist sõltuvate ning kitsendusteta fraasistruktuuri keelte süntaksanalüüsi ülesanne on lahenduv, nii nagu varem vaadeldud regulaarsete ja KV-keelte korralgi. Vastav aktsepteeriv automaat on Turingi masin (kontekstist sõltuvate keelte korral on tegu tõkestatud lindiga Turingi masinaga).

Turingi masina formalism omab teoreetilises informaatikas erilise koha kui vahend, mis lubab näidata ülesannete algoritmiseeritavust. Seepärast käsitleme seda formalismi üldisemast seisukohast lähtudes.

Praktilisest kogemusest teame, et programmide efektiivsus võib olle erinev: sama ülesande lahendamiseks leidub sageli mitu lahendusalgoritmi, mille täitmine nõuab erineval hulgal aega ja ressursse. Mõned arvutiprogrammid töötavad ühel arvutil kiiremini kui teisel, kusjuures mõne teise programmi jaoks võib olla sama suhe vastupidine. Intuitiivselt on aga raske vastata küsimusele, kas leidub ka selliseid ülesandeid, mis pole arvutil üldse lahendatavad. Millised ülesanded on arvutil efektiivselt lahendatavad? Mida üldse mõista efektiivsuse all ja kuidas seda mõõta? Jne.

Eri algoritmide omaduste uurimiseks kasutatakse mitmesuguseid arvutite mudeleid, mis võimaldavad vaatluse alt kõrvale jätta käsitletava küsimuse seisukohalt ebaolulised erinevused arvutite arhitektuurides ja käsusüsteemides, kasutatavate programmeerimiskeelte iseärasused jne.

Turingi masin on abstraktne mudel, mis esitab matemaatiliselt täpselt algoritmi ja arvutusprotsessi mõisted ning võimaldab neid matemaatiliste meetoditega uurida. Selle mudeli esitas 1936.a. inglise loogik Alan Turing. Samal aastal ilmus ameerika matemaatiku Emil Posti artikkel ka teise, Turingi masinaga ekvivalentse arvutamise mudeli - Posti masina - kohta. Hiljem on kasutusele võetud veel teisigi Turingi masinaga samaväärseid mudeleid, nagu näiteks Markovi ahelad, generatiivsed grammatikad jne.

Turingi masinat võib ette kujutada koosnevana arvutusseadmest ja lindist informatsiooni kodeeritud esitamiseks. Lint on mõlemas suunas lõpmatu ning jaotatud diskreetseteks osadeks - pesadeks, millest igaüks võib sisaldada etteantud tähestikus ühe sümboli. Lindiga on seotud lugemis-kirjutamispea, mis võib teatud fikseeritud ajaintervalli jooksul kas lugeda lindilt ühe sümboli või salvestada sinna ühe sümboli. Kirjutamisel kaob varem samas pesas olnud sümbol, s.t ta kirjutatakse üle. Lugemine-kirjutamine on võimalik vaid sellesse lindi pessa, mis on parajasti lugemis-kirjutamispea kohal. Arvutusseade saab ühe ajakavandi jooksul liigutada lugemis-kirjutamispead ühe pesa võrra paremale või vasakule.

Turingi masina arvutusseadme olekute hulk on fikseeritud. Igal ajahetkel võib Turingi masin olla parajasti ühes olekus. Arvutusseadmel on programm, mille järgi ta töötab. Turingi masina programm e. algoritm on käskude hulk, mis määrab arvutusseadme käitumise igas võimalikus olekus. Käsu täitmine tähendab lindilt ühe sümboli lugemist ning sellele järgnevat siirdumist uude olekusse koos lugemis-kirjutamispea liigutamisega või sümboli kirjutamisega lindile. Millisesse olekusse siirdutakse ja kas lugemis-kirjutamispea kirjutab lindile sümboli, sõltub käsu täitmise algul lindilt loetud sümbolist ja masina olekust.

Turingi masina olekute hulgast tõstetakse esile algolek ja üks või mitu lõppolekut. Eeldatakse, et enne masina käivitamist on lindile kirjutatud algandmed (mingi sõna Turingi masina lindi tähestikus), lindil olev sõna pärast masina peatumist lõppolekus on masina töö resultaat. Kui Turingi masina $\mathcal{A}$ lindil on enne töö algust sõna x ning pärast töö lõppu on seal sõna $y = f(x)$, siis öeldkse, et masin $\mathcal{A}$ realiseerib funktsiooni f.

Turingi masina tähtsus põhineb nn. **Turing-Churchi teesil**: *iga efektiivselt arvutava funktsiooni võib realiseerida Turingi masinal* (Igasuguse algoritmilise infotöötluse

võib sooritada Turingi masinal). See väide pole matemaatiliselt tõestatav, sest algoritmilise infotöötluse mõiste ei ole matemaatiline formalism. Küll pole aga viiekümne aasta jooksul seda väidet suudetud ümber lükata. Selliste algoritmiliste protsesside otsimisel, mis poleks teostatavad Turingi masinal, on ilmnenud, et Turingi masin on sobiv formalism mitmesuguste infotöötluse probleemide lahendamiseks, eriti arvutatavuse uurimiseks.

Lisaks eeldame veel täiendavalt (ja seda eeldust järgime kogu aeg ka edaspidi), et passiivses olekus on Turingi masina lugemis-kirjutamispea selle pesa kohal, kus algab "kasulik" info (algandmed või töö resultaat), väljaspool "kasuliku" info piirkonda on lint kogu töö jooksul täidetud tühikutega. Teiste sõnadega, enne masina käivitamist on lindil vaid algandmed ja lugemis-kirjutamispea on valmis argumendi esimese sümboli lugemiseks. Pärast töö lõppu on lindil vaid töö resultaat ning lugemis-kirjutamispea on resultaadi esimese sümboli kohal.

Näide 39. Olgu antud Turingi masin S, mille lindi tähestik koosneb kahest sümbolist $\mathcal{A}_1 = \{\sqcup, |\}$. Masina S olekuteks olgu q_0, q_1 ja q_f, kus q_0 on algolek ja q_f on lõppolek. Turingi masina programm olgu esitatud järgmises tabelis, kus read vastavad masina jooksvale olekule ning veerud vastavad käsu täitmise algul lindilt loetavatele sümbolitele. Tabeli lahtrites on näidatud uus olek, millesse masin siirdub järgmisel ajahetkel ning selle siirdumisega samaaegselt lindile kirjutatav sümbol või sümbol "R" või "L". Viimased tähistavad vastavalt lugemis-kirjutamispea liigutamist paremale või vasakule. (Right, Left).

	$\sqcup$	$\|$
q_0	$\|q_1$	Rq_0
q_1	Rq_f	Lq_1
q_f		

Toodud Turingi masina S töö esitamiseks kirjutame üles igal ajahetkel lindile kirjutatud sümbolid (ilma "kasutute" tühikuteta "kasuliku" info ees ja taga). Lindi "sisu" üleskirjutamisel saadavasse sõnasse lisame masina jooksva oleku tähise selle sümboli ette, mis asub lugemis-kirjutamispea all. Näiteks kui lindil on enne masina S töö algust kolm "kriipsu", siis võib arvutusprotsessi kirjeldada lindi järgmiste olekute (Turingi masina konfiguratsioonide) jadana:

$$
\begin{array}{llll}
q_0 & | & | & | \\
| & q_0 & | & | \\
| & | & q_0 & | \\
| & | & | & q_0 \\
| & | & | & q_1\ | \\
| & | & q_1 & | \ | \\
| & q_1 & | & | \ | \\
q_1 & | & | & | \ | \\
q_1 & \sqcup & | & | \ | \ | \\
q_f & | & | & | \ |
\end{array}
$$

Kui lindi sisu tõlgendada naturaalarvuna, mis võrdub lindil olevate kriipsukeste arvuga, siis võib toodud arvutusprotsessi kokkuvõtlikult esitada seosena $S(3) = 4$. Lihtne on näha, et Turingi masin S realiseerib funktsiooni $s(x) = x + 1$.

□

Ülesanne 3. *Koostada Turingi masinad, mis realiseeriksid funktsioonid*

$$l, k : \mathbb{N}^2 \longrightarrow \mathbb{N}$$

nii, et $l(x, y) = x + y$ *ja* $k(x, y) = x \cdot y$.

Järgnevas tuuakse ära definitsioonid 52 - 54, mis esitavad Turingi masina ja tema funktsioneerimise matemaatilise rangusega.

Definitsioon 52. *Turingi masin on viisik* $\mathcal{A} = (A_t, Q, p, q_0, Q_f)$, *kus*

1. $A_t = \{\sqcup, a_1, \ldots, a_t\}$ - *(lindi) sümbolite lõplik tähestik;*
2. $Q = \{q_0, \ldots, q_m\}$ - *olekute lõplik tähestik;*
3. $p : Q \times A_t \longrightarrow Q \times (A_t \cup \{L, R\})$ - *üleminekufunktsioon;*
4. $q_0 \in Q$ - *lähteolek;*
5. $Q \subseteq Q$ - *lõppolekute hulk.*

Definitsioon 53. $\gamma \in A_t^* \times Q \times A_t^*$ *on Turingi masina* $\mathcal{A}$ *konfiguratsioon.*

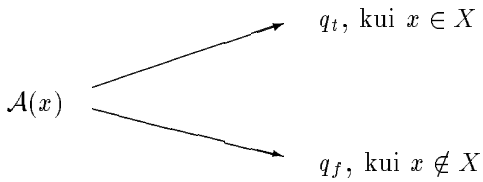
q_0x - *lähtekonfiguratsioon, kus* $x \in A_t^*$;
ry - *lõppkonfiguratsioon, kus* $r \in Q_f$ *ja* $y \in A_t^*$.

Definitsioon 54. $\mathcal{A}(x) = y$, *kui* $q_0x \vdash \gamma_1 \vdash \gamma_2 \vdash \ldots \gamma_n \vdash ry$, *kus*

1. q_0x - *lähtekonfiguratsioon;*
2. ry - *lõppkonfiguratsioon;*
3. $\gamma_i \vdash \gamma_{i+1}$ *parajasti siis, kui kehtib üks seostest (a), (b) või (c):*
   (a) $\gamma_i = uqav$, $\gamma_{i+1} = uaq'v$ *ja* $p(q, a) =< q', R >$;
   (b) $\gamma_i = uaqbv$, $Y_{i+1} = uq'abv$ *ja* $p(q, b) =< q', L >$;
   (c) $\gamma_i = uqav$, $\gamma_{i+1} = uq'bv$ *ja* $p(q, a) =< g', b >$.

Asjaolu, et Turingi masin $\mathcal{A}$ lõpetab argumendi x korral töö olekus r, tähistame $\mathcal{A}(x) \longrightarrow r$. Kui argumendi väärtuse x korral lõpetab Turingi masin oma töö lõpliku arvu taktide jooksul, kirjutame $\mathcal{A}(x) < \infty$, vastasel juhul $\mathcal{A}(x) = \infty$.

Definitsioon 55. *Turingi masinat* $\mathcal{A}(A_t, Q, p, q_0, Q_f)$, *kus* $Q_f = \{q_t, q_f\}$ *nimetatakse hulga* $X \subseteq A_t^*$ karakteristlikuks Turingi masinaks, *kui*



Hulka, mille jaoks leidub karakteristlik Turingi masin, nimetatakse lahenduvaks hulgaks.

Definitsioon 56. *Hulka* $\mathcal{M}(\mathcal{A} = \{x \mid \mathcal{A}(x) < \infty\})$ *nimetatakse Turingi masina* määramispiirkonnaks. *Hulka, mis on mingi Turingi masina määramispiirkonnaks, nimetatakse* genereeritavaks hulgaks.

Näide 40. Järgneva tabeliga esitatav Turingi masin on sõnade hulga

$$X = \{abY \mid Y \in A^*\}$$

karakteristlik Turingi masin. Tähestik $A = \{a, b\}$.

$\mathcal{A}$	a	b	$\sqcup$
q_0	q_1R	q_fb	$q_f\sqcup$
q_1	q_fL	q_tL	q_fL

Järgnevad Turingi masinad pole hulga X karakteristlikeks masinateks, nad on hulka X genereerivateks Turingi masinateks.

$\mathcal{B}$	$\sqcup$	a	b
q_0	q_2R	q_1R	q_2R
q_1	q_2R	q_2R	q_tL
q_2	q_2R	q_2R	q_2R

$\mathcal{C}$	$\sqcup$	a	b
q_0	$q_0\sqcup$	q_1R	q_0b
q_1	$q_1\sqcup$	q_1a	q_tL

□

Järeldus 3. *Iga lahenduv hulk on genereeritav.*

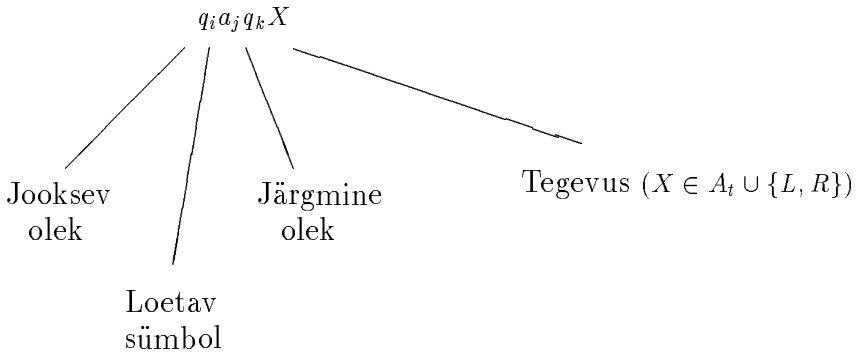
Tõepoolset, hulga karakteristiku Turingi masina saab teisendada seda hulka genereerivaks masinaks. Selleks tuleb kõik üleminekud lõppolekusse q_f asendada üleminekutega uude (mitte lõpp)-olekusse q_{n+1}. Viimase jaoks aga peab laiendama Turingi masina üleminekufunktsiooni s selliselt, et

$$p'(a,a) = \begin{cases} < q', X >= p(q,a), & \text{kui } q\ \{q_0,\ldots,q_n\} = Q \text{ ja } q' \notin Q_f; \\ < q_{n+1}, X >, & \text{kui } p(q,a) =< q_f, X > \text{ja } q_f \in Q_f; \\ < q_{n+1}, R >, & \text{kui } q = q_{n+1}. \end{cases}$$

Turingi teesi erikuju: iga efektiivselt arvutatavate objektide hulk on algoritmiliselt genereeritav.

Turingi masina kodeerimine.

Turingi masina käsk on sõna



Turingi masina üleminekufunktsioon võib olla esitatud programmina e. käskude jadana. Käsud on jadas üksteisest eraldatud sümbolitega ";". Näiteks hulka $X = \{abY \mid Y \in A^*,\ \ A = \{a,b\}\}$ genereeriva Turingi masina C programm oleks järgmine:

$$\begin{array}{lll} q_0 \sqcup q_0\sqcup; & q_0aq_1R; & q_0bq_0b; \\ q_1 \sqcup q_1\sqcup; & q_1aq_1a; & q_1bq_tL; \end{array}$$

Olgu Turingi masina programm $P = x_1x_2\ldots x_n$. Turingi masina kood on sõna $\mathcal{R}(P)$ tähestikus $A_2 = \{\sqcup, |, 0\}$, kus $\mathcal{R}(P) = \mathcal{R}(x_1)\mathcal{R}(x_2)\ldots\mathcal{R}(x_n)$. Seejuures on sümbolite koodid defineeritud järgmisel viisil:

$$\mathcal{R}(a_1) = 0\underbrace{||\ldots||}_{i}0 \quad a_1 \in A_t$$
$$\mathcal{R}(;) = 0\underbrace{|| \ldots ||}_{t+1}0$$
$$\mathcal{R}(L) = 0\underbrace{|| \ \ldots \ ||}_{t+2}0$$
$$\mathcal{R}(R) = 0\underbrace{|| \quad \ldots \quad ||}_{t+3}0$$
$$\mathcal{R}(q_1) = 0\underbrace{|| \quad \ldots \quad ||}_{t+4}0$$

Kokkuvõttes: iga Turingi masinat saab kodeerida, esitades ta sõnana tähestikus $A_2 = \{\sqcup,\ 0,\ |\}$. Samuti on sellist sõna võimalik "töödelda" mõne teise Turingi masina abil. Seega saab Turingi masinat rakendada ka teiste Turingi masinate koodidele.

Näide 41. Leida Turingi masin, mis kopeerib oma sisendi. Masina üleminekufunktsioon on toodud järgnevas tabelis.

Turingi masin, mis kopeerib argumendi väärtuse.

$$\mathcal{U}(x) = x * x,\ x \in \{a, b\}^*,\ A_3 = \{\sqcup, a, b, ^*\}$$

P :	$\sqcup$	a	b	*
q_0	q_0*	q_0R	q_0R	q_1R
q_1	q_2*			
q_2	q_3R	q_2L	q_2L	q_2L
q_3		$q_4\sqcup$	$q_{14}\sqcup$	$q_f\sqcup$
q_4	q_5R			
q_5	q_6a	q_5R	q_rR	q_5R
q_6	q_7a	q_6R		
q_7		q_8L	q_8L	q_8L
q_8		q_9L	q_9L	q_9L
q_9		$q_{10}R$	$q_{11}R$	$q_{12}R$
q_{10}		q_7a	q_7a	
q_{11}		q_7b	q_7b	
q_{12}		$q_{12}*$	$q_{12}*$	$q_{12}L$
q_{13}				q_2a
q_{14}	$q_{15}R$			
q_{15}	$q_{16}b$	$q_{15}R$	q_15R	q_15R
q_{16}	$q_{17}b$		$q_{16}R$	
q_{17}		$q_{18}L$	$q_{18}L$	$q_{18}L$
q_{18}		$q_{19}L$	$q_{19}L$	$q_{19}L$
q_{19}		$q_{20}R$	$q_{21}R$	$q_{22}R$
q_{20}		$q_{17}a$	$q_{17}a$	
q_{21}		$q_{17}b$	$q_{17}b$	
q_{22}		$q_{22}*$	$q_{22}*$	$q_{23}L$
2_{23}				q_2b
q_{24}				
q_{25}				

□

Universaalne Turingi masin.

Definitsioon 57. *Turingi masinat $\mathcal{U}$, mille korral suvalise Turingi masina $\mathcal{R}$ ja argumendi x puhul kehtib seos*

$$\mathcal{U}(\mathcal{R}(\mathcal{U}) * \mathcal{R}(x)) = \mathcal{R}(\mathcal{U}(x)),$$

nimetatakse universaalseks Turingi masinaks.

Teoreem 22. *Ei leidu sellist Turingi masinat, mis suvalise Turingi masina $\mathcal{A}$ ja lähtesõna x korral "tunneb ära", kas masin lõpetab töö või mitte.*

Tõestus. Oletame, et teoreemis kirjeldatud Turingi masin eksisteerib. Olgu selline masin tähistatud sümboliga D. Seega

$$D(\mathcal{R}(\mathcal{A})^*\mathcal{R}(x)) \begin{cases} \longrightarrow q_t, \text{ kui } \mathcal{A}(x) < \infty, \\ \longrightarrow q_f, \text{ kui } \mathcal{A}(x) = \infty. \end{cases}$$

Olgu $\overline{D}^*$ selline Turingi masin, mis rakendatuna $\mathcal{R}(\mathcal{A})$-le, kopeerib kõigepealt oma argumendi ja seejärel "jääb tsüklisse", kui masina D korral on tulemus jaatav, s.t. kui $D(\mathcal{R}(\mathcal{A}) * \mathcal{R}(\mathcal{A})) \longrightarrow q_t$. selline Turingi masin on konstrueeritav näite 41 ja järelduse 3 põhjal.

Seega kehtivad seosed

$$\overline{D}^*(\mathcal{R}(\mathcal{A})) = \begin{cases} \infty, \text{ kui } D(\mathcal{R}(\mathcal{A}) * \mathcal{R}(\mathcal{A})) \longrightarrow q_t \equiv \mathcal{A}(\mathcal{R}(\mathcal{A})) < \infty, \\ < \infty, \text{ kui } D(\mathcal{R}(\mathcal{A}) * \mathcal{R}(\mathcal{A})) \longrightarrow q_f \equiv \mathcal{A}(\mathcal{R}(\mathcal{A})) = \infty. \end{cases}$$

Kokkuvõttes saame, et

$$\overline{D}^*(\mathcal{R}(\mathcal{A})) = \begin{cases} \infty, \text{ kui } \mathcal{A}(\mathcal{R}(\mathcal{A})) < \infty, \\ < \infty, \text{ kui } \mathcal{A}(\mathcal{R}(\mathcal{A})) = \infty. \end{cases}$$

Rakendame nüüd Turingi masinat $\overline{D}^*$ iseenda koodile:

$$\overline{D}^*(\mathcal{R}(\overline{D}^*)) = \begin{cases} \infty, \text{ kui } \overline{D}^*(\mathcal{R}(\overline{D}^*)) < \infty, \\ < \infty, \ (\mathcal{R}(\overline{D}^*)) = \infty. \end{cases}$$

See on ilmne vastuolu. Seega väitevastane oletus Turingi masina D eksisteerimise kohta ei kehti.

□

Järeldus 4. *Leidub ülesandeid, mis pole algoritmiliselt lahenduvad.*

Järeldus 5. *Universaalse Turingi masina määramispiirkond pole lahenduv hulk.*

Tõepoolest, kui universaalse Turingi masina määramispiirkond oleks lahenduv, siis leiduks selline Turingi masin $\mathcal{M}$, et

$$\mathcal{M}(\mathcal{R}(\mathcal{U}) * \mathcal{R}(x)) \begin{cases} \longrightarrow q_t, \text{ kui } \mathcal{U}(\mathcal{R}(\mathcal{U} * \mathcal{R}(x)) < \infty \equiv \mathcal{U}(x) < \infty, \\ \longrightarrow g_f, \text{ kui } \mathcal{U}(\mathcal{R}(\mathcal{U}) * \mathcal{R}(x)) = \infty \equiv \mathcal{U}(x) = \infty. \end{cases}$$

Nagu näha, on masin $\mathcal{M}$ ekvivalentne teoreemi 22 tõestuses kasutatud masinaga D. Kuna sellist masinat ei leidu, siis pole universaalse Turingi masina määramispiirkond lahenduv.

4.2. Rekursiooniteooria

Praktilistes arvutustes (inseneriarvutused, füüsikaliste mõõtmistulemuste töötlemine, majandusanalüüs jne.) kasutatavate arvutusvõtete arv on üpris piiratud.

Vaatleme näitena funktsioonide ligikaudse arvutamise valemeid:

$$e^x \approx \sum_{n=1}^{M} \frac{x^n}{n!} = S_M \quad (\text{ Täpsus } \mid S_M - e^x \mid \; < 10^{-k}),$$

$$\sqrt{x} \approx y_{n+1} = 0,5\left(y_n + \tfrac{x}{y_n}\right).$$

Nendes kasutatavad arvutamisvõtted (operaatorid):

- funktsioonide järjest rakendamine e. *superpositsioon*;
- lähendamine rekursiooni abil, kusjuures kasutatakse sama funktsiooni eelmisi väärtusi;
- *minimeerimine* e. arvutamine seni, kuni on täidetud teatud tingimus.

Nagu järgnevast selgub, on nende operaatorite hulk teatud mõttes piisav, selleks et esitada mistahes arvutatavaid funktsioone. Selleks tuleb nimetatud operaatorid esitada kõigepealt formaalselt.

Definitsioon 58. *Ütleme, et n-kohaline funktsioon f on saadud m-kohalisest funktsioonist g ning n-kohalistest funktsioonidest $h_1, \ldots, h_m$ superpositsiooni operaatori S^{m+1} rakendamise teel $[f = S^{m+1}(g;\ h_1, \ldots, h_m)]$, kui kõikide $x_1, \ldots, x_n$ väärtuste korral kehtib võrdus*

$$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n))$$

Definitsioon 59. *Ütleme, et $(n+1)$-kohaline funktsioon f on saadud n-kohalisest funktsioonist g ning $(n+2)$-kohalisest funktsioonist h rekursiooni operaatori R rakendamise teel $[f = R(g,h)]$, kui kõikide $x_1, \ldots, x_n, y$ väärtuste korral kehtivad võrdused*

$$\begin{cases} f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n), \\ f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{cases}$$

Näide 42. Funktsioonide esitamine rekursioonioperaatori abil:

1. $f(x) = x!$

   Kuna faktoriaal on defineeritud seostega

   $$\begin{cases} 0! = 1 \\ (x+1)! = (x+1) \cdot x!, \end{cases}$$

   saab tema arvutamiseks kasutada rekursioonioperaatorit $f(x) = R(g,h)$, kus $g = 1$ ja $h(y, f(y)) = (y+1) \cdot f(y)$.
2. $f(x,y) = x \cdot y$

   Korrutise omadustest :

   $$\begin{cases} x \cdot 0 = 0 \\ x \cdot (y+1) = x \cdot y + x \end{cases}$$

   järeldub, et $f(x,y) = R(g,h)$, kus $g(x) = 0$ ja $h(x, y, f(x,y)) = f(x,y) + x$.

Kirjeldatud operaatorid näitavad, kuidas saada lihtsamatest funktsioonidest keerukamaid.

□

Definitsioon 60. *Funktsioone*

$\mathbb{0}^n(x_1, \ldots, x_n) = 0$ *(n-kohaline konstantne funktsioon);*
$s(x) = x + 1$*;*
$I^n_m(x_1, \ldots, x_n) = x_m \qquad m \in \{1, \ldots, n\}$

nimetatakse algfunktsioonideks.

Teoreem 23. *Algfunktsioonid on arvutatavad Turingi mõttes.*

Tõestus. Olgu n kodeeritud stringina $\underbrace{0|||\ldots|0}$

1. $\mathbb{0}^n$

	$\sqcup$	0	|
q_0	$q_2 0$	$q_1 \sqcup$	$q_1 \sqcup$
q_1	$q_0 R$		
q_2	$q_f 0$	$q_2 R$	

2. $s(x) = x + 1$

	$\sqcup$	0	|
q_0	$q_f 0$	$q_0 \|$	$q_0 R$

3. I^n_m

Lindi algseis

$$\cdots \sqcup \underbrace{0|||0}_{m} \underbrace{0||\cdots|0}_{x_1} \underbrace{0||\cdots|0}_{x_2} \cdots \underbrace{0||\cdots|0\sqcup}_{x_n} \cdots$$

Skeem:

	$\sqcup$	0	|
q_0	$q_1 R$	$q_0 \sqcup$	
q_1	$q_2 R$		$q_1 \sqcup$
q_2	$q_3 R$	$q_{m=1}$	$q_2 \sqcup$
q_3	$q_4 R$	$q_{m=2}$	$q_3 \sqcup$
$\vdots$	$\ldots$	$\ldots$	$\ldots$
q_{n+1}	$q_{n+2} R$	$q_{m=n}$	$q_{n+2} \sqcup$
$q_{n+2} \equiv q_{m=1} \equiv q_l$	$q_{l+1} R$		
q_{l+1}		$q_{l+2} R$	
q_{l+2}		$q_{l+3} R$	$q_{l+2} R$
q_{l+3}	$q_{l+5} R$	$q_{l+4} \sqcup$	$q_{l+4} \sqcup$
q_{l+4}	$q_{l+3} R$		
q_{l+5}	$q_{l+5} R$	$q_{l+6} L$	$q_{l+6} L$
q_{l+6}	$q_f R$	$q_{l+6} L$	$q_{l+6} L$
$q_{m=i} \equiv q_k$	$q_{k+1} R$		
q_{k+1}	$q_{k+2} R$	$q_{k+2} \sqcup$	
q_{k+2}	$q_{k+2} R$	$q_{k+3} \sqcup$	$q_{k+2} \sqcup$
$\vdots$	$\ldots$	$\ldots$	$\ldots$
q_{k+3i}	$q_{k+3i+2} R$		
q_{k+3i+1}	$q_{k+3i+2} R$	$q_{k+3i+1} \sqcup$	
q_{k+3i+2}	$q_{k+3i+2} R$	$q_1 \sqcup$	$q_{k+3i+2} \sqcup$
$\vdots$	$\ldots$	$\ldots$	$\ldots$

Olekutele $q_{m=i} \equiv q_k$ kuni q_{k+3i+2} vastavad read esinevad funktsiooni I_m^n Turingi masina tabelis iga $i = 2, 3, \ldots, n$ jaoks.

□

Definitsioon 61. *Funktsiooni* $f : \mathbb{N}^n \longrightarrow \mathbb{N}$ *nimetatakse* lihtrekursiivseks *(LRF), kui* f *on algfunktsioon või ta on saadav algfunktsioonidest superpositsiooni ja rekursioonioperaatori abil.*

Järeldus 6. *Iga LRF* $f : \mathbb{N}^n \longrightarrow \mathbb{N}$ *on kõikjal määratud.*

Näide 43. Summa on lihtrekursiivne funktsioon:

$$S(x, y) = x + y$$

$$\begin{cases} S(x, 0) = x \\ S(x, y + 1) = S(x, y) + 1 \end{cases}$$

$$\begin{cases} g(x) = I_1^1(x) \\ h(x, y, z) = z + 1 = s(z) = s(I_3^3(x, y, z)) \\ h(x, y,\ S(x, y)) = S(x, y) + 1 \end{cases}$$

Summa funktsioon operaatortermina (vrdl. Def. 66): $S = R[I_1^1, S^2[s, I_3^3]]$.

□

Definitsioon 62. *Turingi masinate* M_1 *ja* M_2 *kompositsiooniks nimetatakse nende järjest rakendamist:* $M_2(M_1(x))$. *Kompositsiooni operaatori tähis on* $\circ$. *Seega*

$$M_1 \circ M_2(x) = M_2(M_1(x)).$$

Iga kahe ühise sisendtähestikuga Turingi masina

$$M_1 = (A_t, Q_1, p1, q_0', Q_f')$$

ja

$$M_2 = (A_t, Q_2, p2, q_0'', Q_f'')$$

kompositsiooni saab konstrueerida järgmisel teel:

1. nimetada masinate olekud ümber, nii et $Q_1 \cap Q_2 = \emptyset$;
2. kirjutada masina M_2 tabeli read masina M_1 tabelisse viimase rea järele, kustutades eelnevalt masina M_1 lõppolekutele vastavad read (üleminekufunktsioonide $p1$ ja $p2$ ühendamine);
3. saadud tabelis asendada suunamised masina M_1 lõppolekusse suunamistega masina M_2 algolekusse.

Kokkuvõttes võib öelda, et kompositsioon $M_1 \circ M_2$ on Turingi masin

$$M = (A_t, A_1 \cup Q_2, p, q_0', Q_f''),$$

kus üleminekufunktsioon p rahuldab järgmisi tingimusi (pärast olekute ümbernimetamisi):

$$p(q, a) = \begin{cases} p(q, a), \text{ kui } p(q, a) \notin Q_f'; \\ q_0'', \qquad \text{kui } p(q, a) \in Q_f'. \end{cases}$$

Näide 44. Olgu antud Turingi masinad, mis realiseerivad funktsioone $t(x *_\uparrow y) = x \sqcup_\uparrow y$ ja $\epsilon(x * y) = x *_\uparrow y$ (noolega on tähistatud lugemis-kirjutamispea asukoht). Konstrueerime nende Turingi masinate kompositsiooni $\mathcal{N} = \mathcal{E} \circ \mathcal{I}$.

$\mathcal{E}$	$\sqcup$	0	|	$*$
q_0		q_0R	q_0R	q_f*

$\mathcal{I}$	$\sqcup$	0	|	$*$
q_0	q_fR			$q_0\sqcup$

$\mathcal{N}$	$\sqcup$	0	|	$*$
q_0		q_0R	q_0R	q_1*
q_1	q_fR			$q_1\sqcup$

Masina $\mathcal{N}$ korral $\mathcal{N}(x * y) = x \sqcup_\uparrow y$.

Võib aga koostada ka masina $\mathcal{N}' = \mathcal{N} \circ \mathcal{L} = \mathcal{E} \circ \mathcal{I} \circ \mathcal{L}$ nii et $\mathcal{N}'(x * y) = x \sqcup y$. Antud juhul on komponent $\mathcal{L}$ kujul

$\mathcal{L}$	$\sqcup$	0	|
q_0	q_1L	q_0L	
q_1	q_fR	q_1L	q_1L

Viimasel juhul kehtib seos $\mathcal{L}(x \sqcup_\uparrow y) =_\uparrow x \sqcup y$.

□

Näide 45. Turingi masin, mis lindil info "kokku suruks", on esitatav kahe lihtsama Turingi masina kompositsioonina.

$$\mathcal{B} = \mathcal{B}' \circ \mathcal{L}, \text{ kus } \mathcal{B}'(_\uparrow x \sqcup \ldots \sqcup y) = xy_\uparrow$$

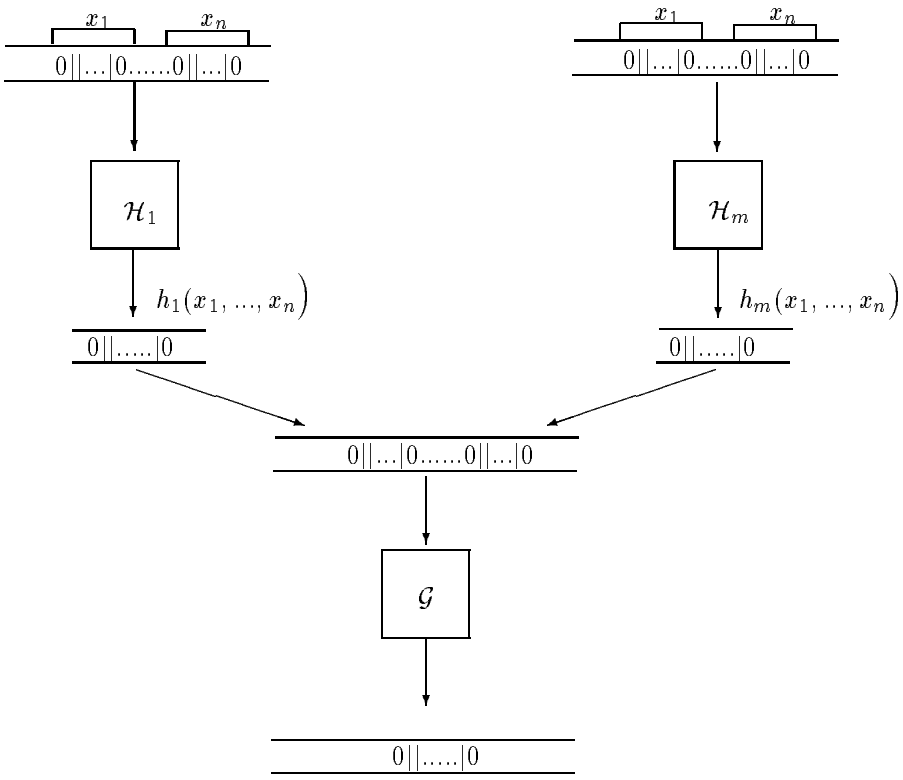
$\mathcal{B}$	$\sqcup$	0	|
q_0	q_1R	q_0R	q_0R
q_1	q_1R	$q_2\sqcup$	$q_6\sqcup$
q_2	q_3R		
q_3	$q_{10}L$	q_4L	q_4L
q_4	q_4L	q_5R	q_5R
q_5	q_50		
q_6	q_7R		
q_7	$q_{12}L$	q_8L	q_8L
q_8	q_8L	q_9R	q_9R
q_9	$q_9\|$		q_1R
q_{10}	$q_{10}L$	$q11R$	$q_{11}R$
q_{11}	q_f0		
q_{12}	$q_{12}L$	$q_{13}R$	$q_{13}R$
q_{13}	$q_f\|$		

$$\mathcal{B}(x \sqcup \ldots y) = xy$$

□

Teoreem 24. *Kui g on m-kohaline arvutatav funktsioon ja $h_1, \ldots h_m$ on n-kohalised arvutatavad funktsioonid, siis on ka funktsioon $f = S^{m+1}[g; h_1, \ldots, h_m]$ arvutatav.*

Tõestus. Eelduse põhjal leiduvad Turingi masinad $\mathcal{G}, \mathcal{H}_1, \ldots, \mathcal{H}_m$, mis realiseerivad funktsioone $g, h_1, \ldots, h_m$.



$$g(h_1(x_1, ..., x_n), ..., h_m(x_1, ..., x_n)) = f(x_1, ..., x_n)$$

Turingi masina, mis realiseeriks funktsiooni f, saab olemasolevatest masinatest konstrueerida järgmiste kompositsioonide abil:

$$\mathcal{J} = \underbrace{\mathcal{R}' \circ \ldots \circ \mathcal{L}'}_{m-1 \; korda} \circ \mathcal{H}_m \circ \mathcal{H}'_{m-1} \circ \mathcal{H}'_{m-2} \circ \ldots \circ \mathcal{H}'_1 \circ \mathcal{G},$$

kus $\mathcal{R}' = \mathcal{R} \circ \mathcal{N}$ ($\mathcal{R}$ on sisendit kopeeriv Turingi masin ja $\mathcal{N}$ on sama, mis näites 44) ja $\mathcal{H}'_i = \mathcal{L} \circ \mathcal{H}_i \circ \mathcal{B}$ $i = 1, 2, \ldots, m$ (Turingi masinad $\mathcal{L}$ ja $\mathcal{B}$ on võetud näidetest 44 ja 45).

Teoreem 25. *Kui g on $n+2$-kohaline arvutatav funktsioon ja h n-kohaline arvutatav funktsioon, siis on $f = R[g, h]$ $n+1$-kohaline arvutatav funktsioon. (Tõestada iseseisvalt).*

Järeldus 7. *Kõik lihtrekursiivsed funktsioonid on arvutatavad.*

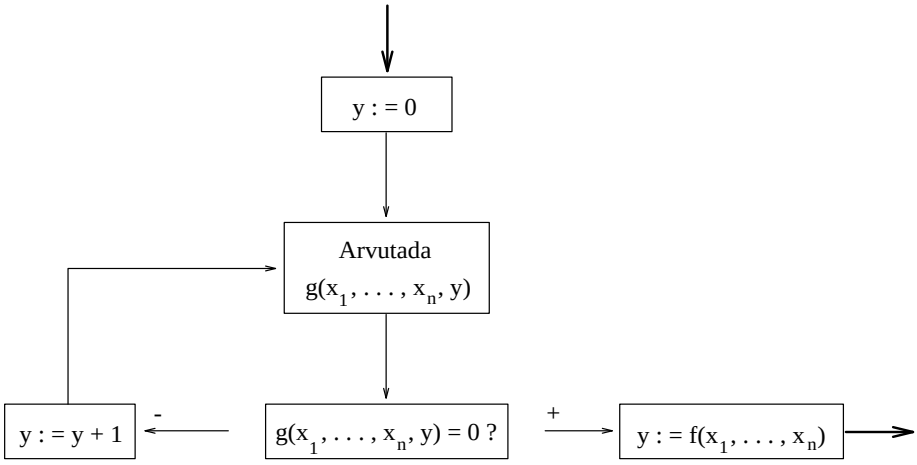
□

Vastupidine ei kehti: arvutatavaid funktsioone on rohkem kui lihtrekursiivseid.

Definitsioon 63. *Ütleme, et n-kohaline funktsioon f on saadud $n+1$-kohalisest funktsioonist g minimeerimisoperaatori μ_y abil ning kirjutame $f = \mu_y[g]$, kui muutujate $x_1, \ldots, x_n$ kõigi väärtuste korral*

$$f(x_1, \ldots, x_n) = \begin{cases} y, \text{kus } y \text{ on vähim selline element, et } g(x_1, \ldots, x_n, y) = 0, \\ \text{kusjuures iga } z < y \text{ korral } g(x_1, \ldots, x_n, z) \text{ on määratud} \\ \text{ja } \neq 0; \\ \text{pole määratud, vastasel juhul.} \end{cases}$$

μ_y-operaatori abil esitatud funktsiooni väärtust võib arvutada vastavalt järgmisele plokkskeemile.



Definitsioon 64. *Funktsioone, mis on saadud algfunktsioonidest superpositsiooni-, lihtrekursiooni- ja minimeerimisoperaatori rakendamise teel, nimetatakse osaliselt rekursiivseteks funktsioonideks.*

Järeldus 8. *Lihtrekursiivsed funktsioonid on osaliselt rekursiivsed.*

Järeldus 9. *Osaliselt rekursiivsed funktsioonid ei pruugi olla kõikjal määratud.*

Põhjusi, miks osaliselt rekursiivne funktsioon võib teatud kohal olla määramata, on kaks:

1. *ei leidu sellist väärtust y, et $g(x_1, \ldots, x_n, y) = 0$;*
2. *leidub väärtus y, nii et $g(x_1, \ldots, x_n, y) = 0$, kuid $g(x_1, \ldots, x_n, z)$ on määramata mingi $z < y$ korral.*

Definitsioon 65. *Kõikjal määratud osaliselt rekursiivseid funktsioone nimetatakse* üldrekursiivseteks funktsioonideks.

Näide 46. Funktsioon $f(x, z) = \mu_y[(|x - z| - \text{sign}(y))]$ on osaliselt rekursiivne funktsioon. Minimeerimisoperaatori argument on lihtrekursiivne funktsioon:

$$\text{sign}(x) = R[\mathbb{O}^0, s(\mathbb{O}^2(x, y))]$$

$$x \dot{-} y = R[I_1^1, R[\mathbb{O}^0, I_1^2](I_3^3)]$$

$$|x - z| = (z \dot{-} x) + (z \dot{-} x)$$

Tulemuseks saame

$$f(x, z) = \begin{cases} 0, \text{kui } |x - z| = 0; \\ 1, \text{kui } |x - z| = 1; \\ \text{määramata, kui } |x - z| > 1. \end{cases}$$

Näiteks pole määratud $f(2, 0)$ (põhjusel 2).

□

Näide 47. Osaliselt rekursiivse funktsiooni $h(x) = \mu_z[f(x, z)]$ väärtus on kohal $x = 2$ määramata, kuigi $f(2, 2) = 0$ (põhjusel 1).

□

Teoreem 26. *Osaliselt rekursiivsete funktsioonide hulk langeb kokku Turingi mõttes arvutatavate funktsioonide hulgaga.*

(Tõestuseta).

□

4.3. Gödeli numbrid

Definitsioon 66. *Operaatortermiks on:*

- *iga funktsioonisümbol;*
- *avaldis kujul $E(t_1, \ldots, t_n)$, kus E on n-kohaline operaator ja $t_1, \ldots, t_n$ on operaatortermid.*

Muid operaatorterme pole.

Operaatortermid esitavad funktsioone. Näiteks funktsiooni $f(x) = x \dot{-} 1$ esitab term $R[0, I_2^3]$.

Funktsioone, mis on esitatavad ainult sümboleid $\mathbb{0}^0, s, I_m^n, S^{m+1}$ ja R sisaldavate operaatortermide abil, nimetatakse lihtrekursiivseteks funktsioonideks. Kui operaatorterm sisaldab ka minimeerimisoperaatori tähise, siis on tegu osaliselt rekursiivse funktsiooniga e. lihtsalt *rekursiivse funktsiooniga*. Rekursiivseid funktsioone, mis on kõikjal määratud, nimetatakse ka *üldrekursiivseteks funktsioonideks*.

Mitmekohalisi rekursiivseid funktsioone võib käsitleda ka ühekohalistena. Selleks toome kõigepealt sisse korteežide järjestuse.

Kahekohaliste korteežide järjekorra moodustamiseks sobib Cantori funktsioon $c(x, y) = 0,5(x + y)(x + y + 1) + x$. Paaridele vastavusse seatavad järjekorranumbrid on

$$c(0,0) = 0, \quad c(0,1) = 1, \quad c(1,0) = 2, \quad c(0,2) = 3, \quad c(1,1) = 4, \ \ldots$$

Leiduvad ka pöördfunktsioonid, mis paari järjekorranumbrile seavad vastavusse selle paari esimese ja teise elemendi

$l(c(x, y)) = x;$
$r(c(x, y)) = y.$

Ülesanne 4. *Leida funktsioonide l ja r analüütiline kuju.*

Kolmekohaliste korteežide kodeerimiseks saab kasutada funktsiooni

$$c^3(x, y, z) = c(c(x, y), z).$$

Dekodeerimiseks sobivad aga funktsioonid

$c_1^3(n) = l(1(n)),$
$c_2^3(n) = r(1(n)),$
$c^3(n) = r(n).$

Analoogiliselt saab kodeerimist laiendada kuitahes pikkade korteežide hulgale. Näiteks i-kohaliste korteežide numeratsioon on esitatav funktsiooniga

$$c^i(x_1, \ldots, x_i) = c(c^{i-1}(x_1, \ldots, x_{i-1}), x_i).$$

Teoreem 27. *Kui superpositsiooni operaatori suhtes kinnine funktsioonide klass sisaldab korteežide kodeerimise ja dekodeerimise funktsioone $c^m, c_1^m, \ldots, c_m^m$, siis selle klassi iga m-kohalise funktsiooni f jaoks leidub samasse klassi kuuluv ühekohaline funktsioon (funktsiooni ühekohaline esindaja) g, nii et muutujate $x_1, \ldots, x_m$ kõigi funktsiooni f määramispiirkonda kuuluvate väärtuste korral kehtib võrdus*

$$f(x_1, \ldots, x_m) = g(c^m(x_1, \ldots, x_m)).$$

Tõestus. Iga $n = c^m(x_1, \ldots, x_m)$ jaoks kehtib seos

$$g(n) = f(c_1^m(n), \ldots, c_m^m(n)).$$

Järelikult saab mitmekohalised funktsioonid asendada ühekohaliste funktsioonidega. Selleks, et ühekohaliste funktsioonide hulgal uurida arvutatavuse probleeme, vajame operaatoreid ühekohaliste funktsioonide genereerimiseks algfunktsioonidest.

Toome järgnevalt mõned operaatorid ühekohaliste funktsioonide konstrueerimiseks.

Definitsioon 67. *Funktsioon $h(x)$ on saadud funktsioonidest f ja g liitmisoperaatori kasutamise teel, kui muutuja x iga väärtuse korral*

$$h(x) = f(x) + g(x).$$

Vastav tähistus on $h = f + g$.

Definitsioon 68. *Funktsioon $h(x)$ on saadud funktsioonidest f ja g kompositsiooni-operaatori kasutamise teel, kui muutuja x iga väärtuse korral*

$$h(x) = f(g(h)).$$

Vastav tähistus on $h = f \cdot g$.

Definitsioon 69. *Funktsioon $h(x)$ on saadud funktsioonist f pööramisoperaatori kasutamise teel, kui muutuja x iga väärtuse korral*

$$h(x) = \mu_z[f(z) - x].$$

Vastav tähistus on $h = f^{-1}$.

Definitsioonis 69 kasutatav lahutamise tehe $f(z) - x$ on määratud vaid juhul, kui $f(z) \geq x$.

Pööramisoperaator on minimeerimisoperaatori teatav ühekohaline analoog. Lihtrekursioonioperaatori lihtsamaks, "ühekohaliseks vormiks" on järgmine *iteratsiooniope-raator.*

Definitsioon 70. *Funktsioon $h(x)$ on saadud funktsioonist f iteratsioonioperaatori kasutamise teel, kui muutuja x iga väärtuse korral*

$$h(x) = \underbrace{f(f(\ldots f}_{x\,korda}(0)\ldots)).$$

Vastav tähistus on $h = \iota f$.

Näiteks, kui $f(x) = 2 + x$, siis $\iota f(x) = 2x$.

Ühekohaliste funktsioonide arvutatavuse määravad kaks Robinsoni teoreemi.

Teoreem 28. *(Robinsoni I teoreem). Kõik ühekohalised lihtrekursiivsed funktsioonid on saadavad algfunktsioonidest $s(x) = x+1$ ja $q(x) = x - \lfloor\sqrt{x}\rfloor^2$ (ruutjäägi funktsioon), kasutades liitmise, kompositsiooni- ja iteratsioonioperaatorit.*[3]

□

Teoreem 29. *(Robinsoni II teoreem). Kõik ühekohalised osaliselt rekursiivsed funktsioonid on saadavad algfunktsioonidest $s(x) = x + 1$ ja $q(x) = x - \lfloor\sqrt{x}\rfloor^2$, kasutades liitmise, kompositsiooni- ja pööramisoperaatorit.*

Funktsiooni gödeliseerimiseks *nimetatakse tema Gödeli numbri väljaarvutamist.*

[3] Avaldis $\lfloor a \rfloor$ tähistab reaalarvu a täisosa.

□

Definitsioon 71. *Funktsiooni h Gödeli number on arv Gh, mis arvutatakse vastavalt järgmisele valemile:*

$$Gh = \begin{cases} 2, kui\ h = s; \\ 3, kui\ h = q; \\ 5^{Gf} * 7^{Gg}, kui\ h = f + g; \\ 11^{Gf} * 13^{Gg}, kui\ h = f \circ g; \\ 17^{Gf}, kui\ h = f^{-1}; \\ 19^{Gf}, kui\ h = \iota f. \end{cases}$$

Sisuliselt kodeerib Gödeli number funktsioonile vastava operaatortermi. Teatud juhtudel võib ühel ja samal funktsioonil olla ka mitu erinevat Gödeli numbrit:

$$Gh = 5^2 * 7^3 = 8575,\ kui\ h(x) = s(x) + q(x)$$

ja

$$Gh = 5^3 * 7^2 = 6125,\ kui\ h(x) = q(x) + s(x).$$

Vaatleme mingit funktsioonide klassi F. Klassi F k-kohaliste funktsioonide alamhulga tähistame sümboliga F^k.

Definitsioon 72. *k+1-kohalist funktsiooni U nimetatakse alamklassi F^k universaalseks funktsiooniks, kui*

1. *iga fikseeritud väärtuse a korral kuulub muutujatest $x_1, \ldots, x_n$ sõltuv funktsioon $U(a, x_1, \ldots, x_k)$ klassi F^k;*
2. *iga funktsiooni $f \in F^k$ korral leidub selline naturaalarv b, et kõikide muutujate $x_1, \ldots, x_n$ korral kehtib võrdus*

$$f(x_1, \ldots, x_k) = U(b, x_1, \ldots, x_n).$$

Kui funktsioonide klass F sisaldab Cantori funktsiooni ja selle pöördfunktsioone, siis saab ka alamklassi F^k universaalse funktsiooni (muidugi vaid juhul, kui selline olemas on) asendada ühekohaliste funktsioonide klassi F^1 universaalse funktsiooniga:

$$U^k(a, x_1, \ldots, x_n) = U^1(a, c^k(x_1, \ldots, x_n)).$$

Teoreem 30. *(Cantori teoreem). Kui asenduste suhtes kinnises kõikjal määratud funktsioonide klassis leidub selline funktsioon g, et iga väärtuse z korral $g(z) \neq z$, siis ei sisalda F universaalset funktsiooni alamklassile F^k ühegi $k = 1, 2, \ldots$ korral.*

Tõestus. Oletame, et mingi $k = 1, 2, \ldots$ korral siiski leidub klassis F hulga F^k universaalne funktsioon U. Siis saab moodustada muutujatest $x_1, \ldots, x_k$ sõltuva funktsiooni

$$f(x_1, \ldots, x_k) = g(U(x_1, x_1, \ldots, x_k)),$$

mis teoreemi eelduse kohaselt kuulub hulka F.

Universaalse funktsiooni definitsiooni põhjal leidub selline arv b, et iga $x_1, \ldots, x_k$ korral kehtiks võrdus

$$U(b, x_1, \ldots, x_k) = f(x_1, \ldots, x_k).$$

Järelikult kehtib see seos ka $x_1 = b$ jaoks ja seega

$$U(b, b, x_2, \ldots, x_k) = f(b, x_2, \ldots, x_k) = g(U(b, b, x_2, \ldots, x_k)).$$

Viimane võrdus on aga vastuolus teoreemi eelduses esitatud nõudega funktsiooni g kohta.

□

Järeldus 10. *Lihtrekursiivsete funktsioonide jaoks konstrueeritud universaalne funktsioon pole lihtrekursiivne funktsioon.*

Tõestus. Lihtrekursiivsed funktsioonid on kõikjal määratud ja sisaldavad funktsiooni $s(x) = x + 1$, mis rahuldab teoreemis 30 funktsioonile g esitatud nõudeid.

Teoreem 31. *Osaliselt rekursiivsete funktsioonide alamhulga universaalne funktsioon on osaliselt rekursiivne.*

□

(Tõestuseta).

Näiteks võib esitada ka osaliselt rekursiivse funktsiooni definitsiooni universaalse funktsiooni kaudu:

$$U(a,x) = \begin{cases} s(x), \text{kui } a = 2; \\ q(x), \text{kui } a = 3; \\ U(b,x) + U(c,x), \text{kui } a = 5^b * 7^c; \\ U(b, U(c,x)), \text{kui } a = 11^c * 13^b; \\ \mu_z[U(b,z) - x], \text{kui } a = 17^b; \\ \text{määramata ülejäänud juhtudel.} \end{cases}$$

Ühekohalist osaliselt rekursiivset funktsiooni, mille Gödeli number on a, tähistame sümboliga φ_a.

Edasi vaatleme, millistel tingimustel võib saada ühe funktsiooni Gödeli numbritest teise funktsiooni Gödeli numbri.

Olgu antud funktsiooni g Gödeli number. Eeldame, et kõikide $x_1, \ldots, x_n$ korral kehtib seos

$$f(x_a, \ldots, x_n) = g(y_1, \ldots, y_m, x_1, \ldots, x_n).$$

Kasutades λ-notatsiooni, võib funktsiooni f esitada kujul

$$f = \lambda x_1, \ldots, \lambda x_n . g(y_1, \ldots, y_m, x_1, \ldots, x_n).$$

Funktsiooni f Gödeli number olgu a. Kas leidub funktsioon

$$\lambda x_1, \ldots, \lambda x_n . \varphi_a^{(n+m)}(y_1, \ldots, y_m, x_1, \ldots, x_n),$$

mille väärtus on a?

Teoreem 32. *(Kleene' s-m-n teoreem). Leidub selline $m+1$-kohaline arvutatav funktsioon S_n^m nii, et suvaliste $a, y_1, \ldots, y_m$ korral kehtib võrdus*

$$\varphi^{(n)}_{S_n^m(a,y_1,\ldots,y_m)} = \lambda z_1, \ldots, \lambda z_n . \varphi_a^{(m+n)}(y_1, \ldots, y_m, z_1, \ldots, z_n).$$

(Tõestuseta).

Teoreem 32 on arvutiteaduses osaliste arvutuste (*mixed computations*) meetodi teoreetiliseks aluseks.

Näiteks $exp(x,y) = x^y$ on arvutatav funktsioon. Vastav operaatorterm on

$$exp = R[S^2[s; \mathbb{0}^0],\ S^3[*; I_1^3, I_3^3]].$$

Siis on s-m-n teoreemi põhjal arvutatavad ka astmefunktsioon ja eksponentfunktsioon

$$as(x) = \lambda . exp(x,a) = x^a$$

ja

$$ex(y) = \lambda y . exp(c,y) = c^y.$$

S-m-n teoreemi järeldus on ka järgmine teoreem. (Täpsema tõestuse jätame siinkohal andmata).

Teoreem 33. *(Kleene' püsipunkti printsiip). Mistahes arvutatava funktsiooni h korral leidub selline naturaalarv u, et kehtib võrdus*

$$\varphi_u = \varphi_{h(u)}.$$

Teoreemi 33 nimetatakse ka rekursiooniteoreemiks.

Funktsioon $\varphi_{h(u)}$ kujutab endast arvutatavate funktsioonide superpositsiooni. Seega, kui mingi funktsiooni määratluses esineb defineeritav funktsioon võrduse mõlemal poolel ja defineeriv avaldis kujutab endast arvutatavate funktsioonide superpositsiooni, siis on ka defineeritav funktsioon arvutatav.

Näiteks on faktoriaal defineeritav võrdusega

$$f(x) = \underline{\text{sign}}(x) + x * f(x \dot{-} 1) * \text{sign}(x), \tag{16}$$

kus

$$\text{sign}(x) = \begin{cases} 0, \text{ kui } x = 0; \\ 1, \text{ kui } x \neq 0. \end{cases}$$

ja

$$\underline{\text{sign}}(x) = \begin{cases} 1, \text{ kui } x = 0; \\ 0, \text{ kui } x \neq 0. \end{cases}$$

Kuna võrduse (16) parem pool kujutab endast arvutatavat funktsiooni, siis on ka funktsioon $f(x) = x!$ arvutatav.

Kleene' püsipunkti printsiibi teise järeldusena vaatleme Rice teoreemi.

Teoreem 34. *Arvutatavate funktsioonide Gödeli numbrite iga mittetriviaalne hulk on mittelahenduv. (Mittetriviaalseks loetakse siin mittetühja ja mitteuniversaalset arvutatavate funktsioonide hulka).*

Tõestus. Olgu H mingi mittetriviaalne arvutatavate funktsioonide hulk. Oletame väite vastaselt, et hulga H Gödeli numbrite hulk H_G on lahenduv. See tähendab, et leidub Turingi masin (e. arvutatav funktsioon) h, nii et

$$h(x) = \begin{cases} 1, \text{ kui } x \in H_G; \\ 0, \text{ kui } x \notin H_G. \end{cases}$$

Selle funktsiooni põhjal on lihtne konstrueerida uus funktsioon:

$$k(x) = \underline{\text{sign}}(h(x)) * \mu_z[h(z) - 1] + \mathbf{sign}(h(x)) * \mu_z[h(z) - 0].$$

Funktsioon $k(x)$ on arvutatav ja kõikjal määratud, kuna hulga H mittetriviaalsuse tõttu on hulgad H_G ja H'_G mõlemad mittetühjad.

Lihtne on näha, et

$$k(x) = \begin{cases} c' \in H'_G, \text{ kui } x \in H_G \text{ e. } \varphi_x \in H; \\ c \in H_G, \text{ kui } x \in H'_G \text{ e. } \varphi_x \notin H, \end{cases}$$

kus H'_G tähistab hulga H_G täiendit.

Vastavalt Kleene' püsipunkti printsiibile leidub selline Gödeli number u, et kehtib võrdus

$$\varphi_u = \varphi_{k(u)}. \tag{17}$$

Kui $\varphi_u \in H$, siis $k(u) = c'$. Kuna aga $c' \in H'_G$, siis $\varphi_{k(u)}$, mis on aga vastuolus võrdusega (17). Analoogiliselt saame näidata ka, et kui $\varphi_{k(u)} \in H$, siis $\varphi_u \notin H$.

□

Järeldus 11. *Pole olemas algoritmi, mis etteantud Gödeli numbri põhjal otsustaks, kas sellele numbrile vastav funktsioon on kõikjal määratud või mitte.*

Näiteks pole arvutatav funktsioon

$$g(x) = \begin{cases} x+1, & \text{kui funktsioon, mille Gödeli number on } x, \\ & \text{omab väärtust, s.t } z(\varphi_x(z) = 1); \\ 0, & \text{vastasel juhul.} \end{cases}$$

Selle põhjuseks on asjaolu, et näiteks funktsioon $h(x) = 2x+2$, mille Gödeli number on 1225, ei oma väärtust 1 ühegi argumendi väärtuse korral. Vastavalt järeldusele 6 ei leidu aga algoritmi, mis teeks kindlaks, kas funktsioon

$$v(x) = \mu_z[h(z) = 1 \ \& \ x = z]$$

on kõikjal määratud või mitte.

Järeldus 12. *Pole olemas algoritmi, mis Gödeli numbrite järgi tunneks ära, kas kaks antud funktsiooni ühtivad või mitte.*

Järeldus 13. *Pole olemas algoritmi, mis suvalise võrrandi $f(x) = 0$ korral "teeks kindlaks", kas võrrand on lahenduv või mitte.*

4.4. lgoritmide ja ülesannete keerukus

4.4.1. Ajaline ja mahuline keerukus

Paljud lahenduvad ülesanded ei ole praktikas siiski arvuti abil lahendatavad, kuna nõuavad selleks liiga palju aega ja/või mälu või muid ressursse. Algoritmi nn. ressursinõudlikkust iseloomustab keerukuse mõiste.

Algoritmi keerukus on arvnäitaja, mis iseloomustab algoritmi tööaega või vajalikku mälu suurust sõltuvalt algandmete mahust. Enamasti kasutataksegi algoritmi hindamisel tema ajalist ning mahulist keerukust. Näiteks Turingi masina M korral iseloomustab tema ajalist keerukust funktsioon $T_M(n)$ - töötaktide arv, mis kulub sisendsõna x töötlemiseks, $n = |x|$. Mahuline keerukus $S_M(n)$ on sisendsõna x töötlemiseks kasutatavate lindi pesade arv. Arvutusseadme teiste mudelite korral mõõdetakse programmi (algoritmi) tööaega vastava mudeli käskude (üleminekute) arvuga ning mahtu antud mudelis kasutatavate mälupesade arvuga. Näiteks magasinmäluga automaadi korral on ajaline keerukus hinnatav üleminekute arvuga, maht aga jooksvalt magasini salvestatavate mitteterminaalide arvuga.

Üldistatult võib öelda, et algoritmi **ajalist keerukust** väljendab funktsioon f, mis igale selle algoritmi järgi lahendatavale konkreetsele ülesandele andmete mahuga n seab vastavusse selle lahendamisel sooritatavate algoritmi sammude arvu $f(n)$.

Algoritmide realiseerimisel saadavate arvutiprogrammide töökiiruse vahetu teoreetiline hindamine on väga keeruline, sõltudes suurest hulgast praktilistest detailidest. Vastavate algoritmide ajalise keerukuse teadmine aga annab lihtsa võimaluse ka nende alusel koostatud programmide võrdlemiseks. Reeglina võib eeldada, et programmi töö aeg on vastava algoritmi ajalise keerukuse kordne $cf(n)$, kus c on konstant. Näiteks saab võrrelda erinevate algoritmide realisatsioone ühes ja samas arvutikeskkonnas, sealhulgas määrata ülesande lahendusaja kasvu sõltuvalt algandmete mahu kasvust (vt. tabel 1). Seejuures pole konkreetset (i-ndat) realisatsiooni iseloomustavat kordajat c_i vaja teadagi.

Tabel 1

Lahendamisaja suhteline kasvamine,
kui algandmete maht suureneb 5-lt 25-le

Programmi töö aeg $cf(n)$	Lahendamisaja suhteline suurenemine $f(25)/f(5)$
c_1	1
$c_2 \log n$	2
$c_3 n$	5
$c_4 n \log n$	10
$c_5 n^2$	25
$c_6 n^3$	125
$c_7 2^n$	1048576

Algoritmi ajalise keerukuse hinnang määrab ka algoritmi praktilise rakendatavuse piiri: väga kiiresti kasvava keerukusfunktsiooniga algoritmi tasub realiseerida väikesemahuliste ülesannete ringi jaoks, kuna suuremate ülesannete lahendusaeg osutub praktiliselt lõpmatuks (vt. tabel 2).

Tabel 2

Erineva keerukusega programmide ajalised piirid

Keerukus (mikrosek.)	Suurim ülesanne, mille lahendamise aeg $<$ 1 sek.	Suurim ülesanne, mille lahendamise aeg $<$ 1 päev	Suurim ülesanne, mille lahendamise aeg $<$ 1 aasta
n	$n = 1000000$	$n = 86400000000$	$n = 31530000000000$
$n \log_2 n$	$n = 62746$	$n = 2755147514$	$n = 798160978500$
n^2	$n = 1000$	$n = 293938$	$n = 5615692$
n^3	$n = 100$	$n = 4421$	$n = 31593$
2^n	$n = 19$	$n = 36$	$n = 44$
$n!$	$n = 9$	$n = 14$	$n = 16$

Algoritmide ajalise keerukuse juures pakub suuremat huvi just vastavate funktsioonide käitumine algandmete mahu piiramatul kasvamisel, s.t. argumendi n küllalt suurte väärtuste korral - nn. **asümptootiline hinnang**. Nagu ka tabelist 1 näha, funktsioonid kujul $cn \log n$, mis erinevad ainult kordaja c poolest, kasvavad suhtelises mõttes kõik ühtemoodi (seejuures sõltumata kordajast c). Analoogiliselt, kõik funktsioonid kujul $c2^n$ kasvavad omakorda ühetaoliselt, kuid oluliselt kiiremini kui eespool-nimetatud. Seega funktsioonid kujul $cn \log n$ on asümptootiliselt ühe ja sama hinnanguga, öeldakse, et nad on $\mathcal{O}(n \log n)$; teise nimetatud klassi kuuluvad aga funktsioonid kujul $c2^n$, nende kohta öeldakse, et nad on $\mathcal{O}(2^n)$. Üldiselt on (positiivne) funktsioon $f(n)$ asümptootilise hinnanguga $\mathcal{O}(g(n))$, kui külalt suurte argumendi väärtuste korral $f(n) < cg(n)$, kus $c > 0$ on konstant. Näiteks mitte ainult $126n \log n$ on $\mathcal{O}(n \log n)$, vaid ka $5n \log n + 100 \log n$ on $\mathcal{O}(n \log n)$. Tõepoolest, leidub konstant c, nii et teatud väärtusest $n = N$ alates $5n \log n + 100 \log n < cn \log n$:

$$(5n + 100) \log n < cn \log n,$$

$$5n + 100 < cn,$$

$$100 < n(c - 5).$$

Seega võib võtta $c = 6$ ja $N = 100$. Järgnevas esitatakse $\mathcal{O}$-mõiste rangem määratlus.

Definitsioon 73. *Olgu f ja g naturaalarvuliste argumentidega funktsioonid. Siis f on $\mathcal{O}(g)$ parajasti siis, kui leiduvad $c > 0$ ja $N > 0$ nii, et $|f(n)| \leq c|g(n)|$ iga $n > N$ korral.*

Teoreem 35. *Olgu f ja g naturaalarvuliste argumentidega ja positiivsete väärtustega funktsioonid. Siis f on $\mathcal{O}(g)$ parajasti siis, kui leidub $c > 0$ nii, et $f(n) \mid \leq cg(n)$ iga naturaalarvu n korral.*

Tõestus. Piisavus on ilmne.
Tarvilikkus. Kui f on $\mathcal{O}(g)$, siis leiduvad c ja N vastava definitsiooni kohaselt, s.t.

$$f(n) \leq cg(n) \quad \forall n, n > N.$$

Olgu

$$d = \max_{0<n<N}\{f(n)/g(n)\},$$

siis

$$f(n) \leq dg(n) \quad \forall n, 0 < n \leq N.$$

Kokkuvõttes,

$$f(n) \leq c'g(n) \quad \forall n, n > 0,$$

kus $c' = max\{c, d\}$.

Edasises vaadeldavad algoritmide ajalised keerukused ja nende hinnangud vastavad kõik ülaltoodud teoreemi eeldustele, seega võime piirduda määratlusega: f on $\mathcal{O}(g)$ parajasti siis, kui leidub $c > 0$ nii, et $f(n) \leq cg(n)$ iga naturaalarvu n korral.

Paneme tähele, et $\mathcal{O}$-relatsioon ei tähenda mitte ühe funktsiooni väärtuste tõkestatust teise funktsiooni väärtustega, vaid nende funktsioonide väärtuste suhte (funktsioonide suhtelise kasvu) tõkestatust. Kui f on $\mathcal{O}(g)$, siis järelikult leidub $c > 0$, nii et $f(n)/g(n) < c$, s.t. funktsiooni f suhteline kasv funktsiooni g suhtes on tõkestatud; teiste sõnadega, argumendi n piiramatul kasvamisel $f(n)/g(n)$ ei kasva piiramatult. Kuna on tegemist tõkestatusega "ülalt", siis saab taolisi funktsioonide asümptootilisi võrdlemisi teha suvalise "liiaga". Näiteks funktsiooni $3n^2+1$ on nii $\mathcal{O}(n^2)$ kui ka $\mathcal{O}(n^3)$ ja $\mathcal{O}(n^4)$ ning isegi $\mathcal{O}(2^n)$. On ju kõik suhted $(3n^2+1)/n^2, (3n^2+1)/n^3, (3n^2+1)/n^4$ ja $(3n^2+1)/2^n$ tõkestatud ülalt (näiteks arvuga 5), kui $n \geq 1$.

Loomulikult pakuvad rohkem huvi võimalikult täpsed (ja samas ka võimalikult lihtsad) hinnangud. Eriti täpselt iseloomustab mingi funktsiooni f kasvamist selline funktsioon g, et f on $\mathcal{O}(g)$ ja g on $\mathcal{O}(f)$ (sellisel puhul öeldakse, et f on $\Theta(g)$). Nimelt osutub siis funktsiooni f suhteline kasv tõkestatuks ka altpoolt: kui f on $\Theta(g)$, siis

$$f \text{ on } \mathcal{O}(g) \Longrightarrow f/g < c,$$

$$g \text{ on } \mathcal{O}(f) \Longrightarrow g/f < c',$$

kus c ja c' on positiivsed konstandid. Järelikult

$$(1/c') < f/g < c.$$

Teiste sõnadega, kui f on $\Theta(g)$, siis funktsiooni f kasvamine funktsiooni g suhtes on küll piiratud, kuid samal ajal ei jää f väärtused g väärtustest ka lõpmatult palju maha. Nii on ülal näitena vaadeldud funktsioonide suhete korral vaid esimene neist alt tõkestatud: $3 < (3n^2+1)/n^2 < 5$. Seega $3n^2+1$ on $\Theta(n^2)$, kuid $3n^2+1$ ei ole $\Theta(n^3)$.

Algoritmi ajalise keerukuse asümptootilisel hindamisel on kasulik teada $\mathcal{O}$-relatsiooni järgmisi omadusi.

1. Iga $k > 0$ korral, kf on $\mathcal{O}(f)$. Konstantsed kordajad ei mängi seega mingit rolli. Erijuhtudel, f on $\mathcal{O}(f)$ ja k (s.t. konstantne funktsioon) on $\mathcal{O}(1)$;

2. Kui f on $\mathcal{O}(g)$ ja h on $\mathcal{O}(g)$, siis $(f+h)$ on $\mathcal{O}(g)$. Erijuhul, kui f on $\mathcal{O}(g)$, siis $(f+g)$ on $\mathcal{O}(g)$;
3. Kui f on $\mathcal{O}(g)$ ja g on $\mathcal{O}(h)$, siis f on $\mathcal{O}(h)$;
4. n^r on $\mathcal{O}(n^s)$, kui $0 \leq r \leq s$;
5. Kui p on d-astme polünoom, siis p on $\Theta(n^d)$. Näiteks $7n^5 + 2n^2 - 1$ on $\Theta(n^5)$;
6. Kui f on $\mathcal{O}(g)$ ja h on $\mathcal{O}(r)$, siis $(f \times h)$ on $\mathcal{O}(g \times r)$;
7. n^k on $\mathcal{O}(b^n)$, kui $b > 1, k \geq 0$. Näiteks n^5 on $\mathcal{O}(2^n)$;
8. $\log_b n$ on $\mathcal{O}(n^k)$, kui $b > 1, k > 0$. Erijuhul, $\log n$ on $\mathcal{O}(n)$.
9. $\log_b n$ on $\Theta(\log_d n)$ iga $b, d > 1$ korral;
10. $\sum_{k=1}^{n} k^r$ on $\Theta(n^{r+1})$.

Omadus 5 võimaldab anda lihtsaid ja täpseid hinnanguid kõigile polünoomina avalduvatele keerukusfunktsioonidele. Algoritmi, mille ajaline keerukus on $\mathcal{O}(n^d)$, kus d on täisarv, nimetatakse **polünomiaalse keerukusega algoritmiks**. Sellised algoritmid moodustavad väga tähtsa klassi, kuna ülejäänud (nendest ajaliselt keerukamad) algoritmid osutuvad vähegi mahukamate algandmete puhul juba lootusetult aeglasteks. Ülesandeid, mille jaoks ei ole teada polünomiaalse keerukusega algoritmi, nimetataksegi **raskelt lahenduvateks ülesanneteks**.

Üllataval kombel leidub hulganisti selliseid raskelt lahenduvaid ülesandeid, mille kohta ei ole õnnestunud veel tõestada, et nad oleksid mittepolünomiaalsed. Teiste sõnadega, on olemas palju ülesandeid, mille jaoks ei ole leitud polünomiaalse keerukusega algoritme ega ole suudetud ka tõestada, et neid ei leidugi. Raskelt lahenduva ülesande näitena võiks nimetada nn. **seljakotiülesannet:** antud n eseme hulgast valida välja selline esemete komplekt, mille koguhind oleks võimalikult suur, kuid mille kogukaal ei ületaks etteantud konstanti (seljakoti lubatud kaalu). Seljakotiülesande näol on tegemist üsnagi tüüpilise raskelt lahenduva ülesandega, kus lahenduse saab leida algandmete hulga kõigi alamhulkade läbivaatamise teel.

Algoritmi **keskmiseks ajaliseks keerukuseks** $A(n)$ nimetatakse operatsioonide suurimat arvu, mida tuleb sooritada keskmiselt ühe konkreetse (andmemahuga n) ülesande lahendamiseks.

Algoritmi **ajaliseks keerukuseks halvimal juhul** $W(n)$ nimetatakse operatsioonide suurimat arvu, mida tuleb sooritada konkreetse (andmemahuga n) ülesande lahendamiseks.

Näiteks sorteerimata nimistust otsimise algoritmil $A(n) = n/2$ ja $W(n) = n$. Antud juhul nende funktsioonide asümptootilised hinnangud langevad kokku: mõlemad on $\Theta(n)$.

Seos "g on $\mathcal{O}(n)$" tähendab ühtlasi seda, et funktsiooni f kasvamine on tõkestatud altpoolt funktsiooni g kasvuga. Sel puhul öeldakse, et f on $\Omega(g)$.

Kui iteratiivse algoritmi ajalise keerukuse hindamisele leitakse tsükli(te) kordamise käigus sooritatavate põhioperasioonide arv, siis rekursiivse algoritmi täitmiseks kuluv aeg avaldatakse rekurrentse võrrandina. Näiteks algoritmi puhul, kus lähteülesanne (mahuga n) jaotatakse kaheks alamülesandeks (mahtudega $n/2$), mõlemad lahendatakse samal meetodil ja seejärel kogu ülesande lahenduse saamiseks kulub veel n sammu (alamülesannete lahenduste töötlemiseks), avaldub lahendusaeg $T(n)$ järgmise rekurrentse võrrandina:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n \text{ (iga } n \geq 2 \text{ korral) ja } T(1) = 0.$$

Rekurrentsete võrrandite lahendamise erivõtete kõrval saab paljudel juhtudel rekursiivse algoritmi rekurrentselt avaldatud lahendusaega hinnata järgmist põhiteoreemi kasutades.

Teoreem 36. *Olgu $a \geq 1$ ja $b > 1$ konstandid ja $f(n)$ funktsioon ning olgu $T(n)$ mittenegatiivsete n väärtuste korral defineeritud valemiga*

$$T(n) = aT(\lfloor n/b \rfloor) + f(n).$$

Siis

1. *$T(n)$ on $\Theta(n^{\log_b a})$, kui $f(n)$ on $\mathcal{O}(n^{\log_b a - \epsilon})$ mingi positiivse konstandi ϵ korral;*
2. *$T(n)$ on $\Theta(n^{\log_b a} \log n)$, kui $f(n)$ on $\Theta(n^{log_b a})$;*
3. *$T(n)$ on $\Theta(f(n))$, kui $f(n)$ on $\Omega(n^{\log_b a + \epsilon})$ mingi positiivse konstandi ϵ korral ning kui $af(n/b) \leq cf(n)$ mingi konstandi $c > 1$ ja kõigi küllalt suurte n väärtuste korral.*

(Käesolevas me selle teoreemi tõestust ei esita).

4.4.2. Ülesannete keerukuse klassid

Mingit tüüpi ülesannete keerukust kirjeldab lahendamiseks vajaliku aja ja mahu ülemine ja alumine hinnang.

Ülemiseks hinnanguks sobib mingi ülesannet lahendava algoritmi keerukus. Alumise hinnangu võib saada ülesande täpsema uurimise teel või mingi tuntud keerukusega ülesande redutseerimisega uuritavale ülesandele. Ülemise ja alumise hinnangu kokkulangemisel saame täpse hinnangu.

Olulise klassi moodustavad ülesanded, mille keerukuse ajaline hinnang on polünomiaalne: $\mathcal{O}(n^k)$, kus k on mingi konstant. Selliste ülesannete klassi tähistatakse P (polynomial).

Sellised on, näiteks, andmetöötlusülesanded failidega tööks.

Paljud kombinatoorsed ülesanded omavad suuremat keerukust. Vaatleme lausearvutuse valemi tõestuse probleemi:

teha kindlaks, kas lausearvutuse valemis esinevatele muutujatele saab selliselt omistada väärtusi tõene ja väär, et valem oleks tõene.

Valemi, s.t. ülesande lähteandmete keerukust võime hinnata temasse kuuluvate muutujate arvuga n. Neile muutujaile saab omistada tõeväärtusi 2^n erineval viisil. Iga selline omistus, sõltumatult teistest, võib valemi tõeseks muuta. Seega, kui valem ei ole tõene ühegi väärtuste komplekti jaoks, tuleb läbi vaadata 2^n juhtu. Selle ülesande keerukuse teadaolev hinnang ongi eksponentsiaalne: $\mathcal{O}(2^n)$.

Teisest küljest, kui arvutamine toimub "oraakli" abil, kes oskab ette öelda sobiva väärtuste komplekti, siis selle kontrollimiseks, kas antud väärtuste korral valem on tõene, läheb vaja aega ainult võrdeliselt valemi pikkusega, aga muutujate arvu n suhtes - polünomiaalselt. Selle ülesande lahendamist võib ette kujutada ka kui järkjärgulist mittedetermineeritud valiku tegemist iga muutuja väärtuse määramiseks. Sellise ülesande keerukust nimetatakse **mittedetermineeritult polünomiaalseks** ja tähistatakse NP (nondeterministic polynomial).

Vaadeldud ülesandega on keerukuse poolest sarnased paljud kombinatoorsed ülesanded. Näiteks Hamiltoni tee leidmine graafil (see on tee, mis läbib iga tipu täpselt üks kord) või lühima tsükli leidmine, mis läbiks kõik graafi tipud, nn. rändkaupmehe ülesanne. Ka nende puhul on võimaliku lahendi, s.t. tee sobivuse kontroll polünomiaalse keerukusega, kuid sellise tee koostamiseks ei ole teada plünomiaalse keerukuse algoritmi. Nende ülesannete omapära seisneb veel selles, et nad on üksteisele taandatavad polünomiaalse keerukusega algoritmi abil. Küllalt lihtne on leida viise, kuidas ülesandeid graafil esitada lausearvutuse ülesannetena, kodeerides graafe ja teid kahendkoodidega nii, et andmemaht kasvab ainult polünomiaalselt. Ka vastupidine kodeering osutub võimalikuks. Näiteks on viimatinimetatud ülesanded taandatavad ka seljakotiülesandele ja vastupidi. Selliseid ülesandeid on leitud palju. Need moodustavad nn. NP täielike ülesannete klassi, mida tähistatkse NPC (NP complete). Mingi ühe NPC ülesande lahendamise algoritm lahendab ka kõiki teisi selle klassi ülesandeid (andmete ümberkodeerimise hinnaga, mis aga on kõigest keerukusega P).

On teada NP ülesandeid, mille kohta pole aga teada, kas nad on NPC. Näiteks ülesanne : "Kas arv on algarv?" Samuti aga ülesanne: "Kas arv on mittealgarv?" Viimane on ilmselt NP, sest teades mingit arvust väiksemat ja 1-st suuremat jagajat, saab

algarvsust lihtsalt kontrollida. Tegelikult ka esimene neist on NP. Kummagi ülesande kohta pole aga teada, kas nad on NPC.

On teada ülesandeid, mis on keerukamad kui NPC. Näiteks selline on $N \times N$ suurusega laual mängitav kabemäng. Graafil on selliseks tõkestusülesanne. Graafi iga kaar on värvitud ühega kolmest värvist. Osa tippe on võidutipud mängijale A või mängijale B. Need kaks mängijat teevad kordamööda käike ja püüavad jõuda võidutippu. Käiguks on liikumine mööda ühevärvilist teed. Selle mängu jaoks võiduka käigu leidmine nõuab üldjuhul aega K^n, kus n on graafi tippude arv. See on eksponentsiaalne alumine hinnang!

Esineb ka hüpereksponentsiaalse keerukusega ülesandeid. Näiteks sellised on kõrgemat järku loogika valemite tõesuse ülesanded, mis võivad omada ajalist keerukust

, kus k on määratud algandmetega, s.o. uuritava valemiga.

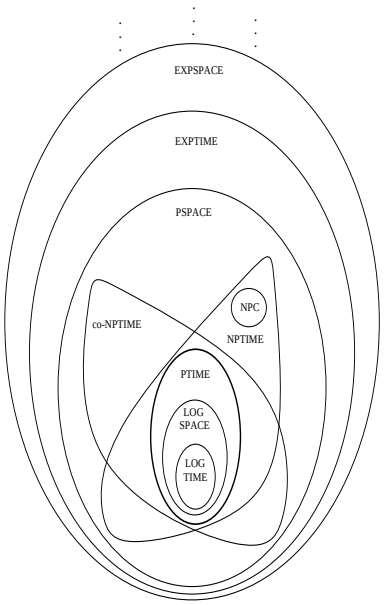
Ajalise ja mahulise keerukuse vahekord

Üldiselt on teada, et ülesannete mahuline keerukus (mäluvajadus) kasvab aeglasemalt kui ajaline keerukus. Näiteks ülesannet, mille ajaline keerukus on $T(n)$, saab lahendada (Turingi masinal) mälumahuga $\mathcal{O}(T(n)/\log T(n))$.

On teada ka ülesandeid, mille mahulise keerukuse alumine hinnang on eksponentsiaalne. Kuid juba polünomialsete (rohkem kui lineaarsete) mahuhinnangutega ülesannete klass, mida tähistatakse PSPACE on keerukate ülesannete klass, sest on teada, et NP $\subseteq$ PSPACE. Analoogiliselt NPC-le on teada ka PSPACE-täielikke ülesandeid. Ülesannete liigitus nende keerukuse järgi on võetav kokku järgmiselt:

LOGTIME $\subseteq$ LOGSPACE $\subseteq$ PTIME $\subseteq$ NPPSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE $\subseteq$ EXPTIME ...

Täpsema ettekujutuse annab järgmine joonis:



Konkreetsete ülesannete ja algoritmide keerukus

Elemendi otsimine n-elemendilisest massiivist: tuleb teha valik n võrdse võimaluse seast, selleks on vaja $\log_2 n$ bitti infot. Seda saab $\log_2 n$ võrdlustehtega. See annab algoritmi keerukuse alumise hinnangu ja ülesande keerukuse täpse hinnangu.

Järjestamine: järjestada kasvavas järjekorras n arvu. Minimaalne keerukus: n korda lahendada elemendi koha leidmise ülesanne. (Viimane on sama, mis elemendi otsimine n elmendi seast). Seega keerukus on $n \log_2 n$.

Sõna otsimine tekstist: leida, kas string x sisaldub stringis y;
1970. a. (Morris, Pratt) algoritm keerukusega $\mathcal{O}(|x| + |y|)$;
Boyer, Moore'i algoritm annab keskmise arvutuskeerukuse $\mathcal{O}(|y| \log_2 |x|/|x|)$, kuigi ülemine hinnang samuti $\mathcal{O}(|x| + |y|)$;
1980. a. (Galil, Seiferas) algoritm, mis kasutab ainult $\mathcal{O}(\log_2 |x|)$ lisamälupesa.

Maatriksite korrutamine: definitsiooni põhjal koostatud algoritmi ajaline keerukus on $\mathcal{O}(n^3)$. Teisi algoritme;

1969. a. (Strassen) $\mathcal{O}(n^{\log_2 7} \cong \mathcal{O}(n^{2.8007})$
1978. a. (Pan) $\mathcal{O}(n^{2.795})$
1980. a. oli teada juba $\mathcal{O}(n^{2.49})$

Alumine hinnang on $\mathcal{O}(n^2)$, kuid täpne pole teada. Kuigi juba $\mathcal{O}(n^{2.49})$ puhul korraja $n^{2.49}$ ees on suur, s.t. väikeste maatriksite jaoks pole sellistel algoritmidel mõtet.

Kivimäng (pebble-game): kivide nihutamine graafil teatud reeglite järgi nii, et käia läbi kõik tipud. S - kivikeste arv, s.o. mälumaht. T - vajalik sammude arv - ajahinnang. Sellele ülesandele taandub avaldise arvutamine, kus S on vajalik registrite (mälupesade) arv. Sellega seotud ülesanne: antud graafi ja S järgi teha kindlaks, kas S-st piisab G kõigi tippude läbimiseks. Kivimängu reeglite järgi on PSPACE-täielik (Gilbert, 1979).

Seega on arusaadav, miks kiiret ja optimaalset mälujaotusalgoritmi pole õnnestunud leida.

On teada aja ja mälumahu seoseid.

1975. a. (Hopcroft) n tipuga graafi läbimiseks piisab $\mathcal{O}(n/\log_2 n)$ mälupesa.

1978. a. (Van Leeuwen jt.) graaf, millel mälumahu S vähendamine ühe pesa võrra suurendab aega $T = \mathcal{O}(n)$-lt $T = \mathcal{O}(exp(n^{1/4} \log_2 n))$-le.

1979. a. (Lenhauer, Tarjan) mistahes n-tipulist graafi saab läbida mäluga

$$n/\log_2 n \leq S \leq n$$

ja ajaga

$$T = S \cdot exp \;\; exp\mathcal{O}(n/S).$$

On teada graafe, kus see hinnang on täpne.