НЕБОЈША ИКОДИНОВИЋ

ТЕОРИЈА АЛГОРИТАМА

(-- БЕЛЕШКЕ --)

ФЕБРУАР 2019.

Увод

Појам алгоритма[1] или ефективне процедуре дуго је био присутан у математици без прецизне дефиниције. Неформално, под алгоритмом се подразумева *механички изводљив поступак, дефинисан коначним бројем инструкција, који се изводи корак по корак над коначним скупом полазних података, при чему је сваки корак недвосмислено дефинисан, и завршава се у коначним временским и просторним оквирима.*

Еуклидов алгоритам. Природно је причу о алгоритмима започети једним од најпознатијих и најстаријих нетривијалних алгоритама. Реч је о чувеном Еуклидовом алгоритму за одређивање највећег заједничког делиоца два природна броја. Следеће две једнакости на сажет и елегантан начин описују Еуклидов алгоритам:

(1) $\mathbf{NZD}(m,0) = m, m \geqslant 0$

(2) $\mathbf{NZD}(m,n) = \mathbf{NZD}(n, m \bmod n), m \geqslant 0, n > 0.$

Штавише, једнакости (1) и (2) дефинишу јединствену функцију $\mathrm{NZD} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (која сваком пару природних бројева додељује њихов највећи заједнички делилац) и истовремено представљају правила за израчунавање њених вредности.

X Хилбертов проблем. Линеарна Диофантова једначина јесте једначина облика $ax + by = c$, где су a, b и c задати природни бројеви, чија решења x и y тражимо у скупу целих бројева. Када су нам познати коефицијенти a, b и c једноставно је утврдити да ли ова једначина има целобројна решења или их нема:

$$\exists x \in \mathbb{Z}\, \exists y \in \mathbb{Z}\, (ax + by = c) \;\Leftrightarrow\; \mathrm{NZD}(a,b) \mid c.$$

Алгоритам за проверу услова са десне стране истовремено представља и алгоритам којим се утврђује да ли било која задата линеарна Диофантова једначина има целобројна решења или их нема.[4]

Ситуација се знатно компликује када посматрамо општи проблем: Ако је $P(x_1, \ldots, x_k)$ произвољан полином са целобројним коефицијентима, испитати да ли једначина $P(x_1, \ldots, x_k) = 0$ има целобројна решења или их нема.[5]

На II светском конгресу математичара, који је одржан 1900. године у Паризу, Хилберт је поставио следећи проблем, данас познат као X Хилбертов проблем: *За дату Диофантову једначину са целобројним коефицијентима и било којим бројем непознатих, измислити* ***поступак*** *којим се може одлучити, користећи коначан број операција, да ли та једначина има или нема решења.* У то време се о *поступцима* расправљало само на интуитивном нивоу,

[1] Реч *алгоритам* долази од латинизованог имена арапског математичара Абу Џафар Мухамед Ибн Муса **Ал Хорезмија** који је у IX веку дао велики допринос математици својим делом Hisab **al džabr** val mukabala. (Реч *алгебра* долази од *ал џабр*.) Почетком XII века, једна Ал Хоризмијева књига је преведена на латински под насловом Algoritmi de numero indorum (Ал Хорезми о индијској вештини рачунања). Тада је у Европу стигао савремени позициони систем записивања бројева.

$\mathrm{NZD}(942,444) = ?$
$942 = 2 \cdot 444 + 54$
$\mathrm{NZD}(942,444) = \mathrm{NZD}(444,54)$
$444 = 8 \cdot 54 + 12$
$\mathrm{NZD}(444,54) = \mathrm{NZD}(54,12)$
$54 = 4 \cdot 12 + 6$
$\mathrm{NZD}(54,12) = \mathrm{NZD}(12,6)$
$12 = 2 \cdot 6 + 0$
$\mathrm{NZD}(12,6) = \mathrm{NZD}(6,0) = 6$
$\boxed{\mathrm{NZD}(942,444) = 6}$

$\mathrm{NZD}(444,942)$
$= \mathrm{NZD}(942, 444 \bmod 942)$ Правило (2)
$= \mathrm{NZD}(942,444)$
$= \mathrm{NZD}(444, 942 \bmod 444)$ Правило (2)
$= \mathrm{NZD}(444,54)$
$= \mathrm{NZD}(54, 444 \bmod 54)$ Правило (2)
$= \mathrm{NZD}(54,12)$
$= \mathrm{NZD}(12, 54 \bmod 12)$ Правило (2)
$= \mathrm{NZD}(12,6)$
$= \mathrm{NZD}(6, 12 \bmod 6)$ Правило (2)
$= \mathrm{NZD}(6,0)$
$= \mathrm{NZD}(6,0) = 6$ Правило (1)

[4] Уколико нека линеарна Диофантова једначина има решења, користећи Еуклидов алгоритам можемо наћи бар једно решење. Испоставља се да када пронађемо једно решење, лако је описати и скуп свих решења.

[5] На пример, једначина $x_1^2 + x_2^2 = x_3^2$ има целобројна решења, док $15x_1^2 - 7x_2^2 = 9$ нема целобројна решења.

па није потпуно јасно шта је Хилберт тачно имао на уму. Ипак, без опасности да много погрешимо, наведени проблем бисмо могли да преформулишемо на савремени језик: Да ли се може написати програм (у неком програмском језику) чији би улази били Диофантове једначине и који би утврђивао да ли задата једначина има или нема целобројна решења (уз претпоставку да рачунар који извршава тај програма поседује меморијски простор колики год да му је потребан и да у коначном времену, колико год да оно траје, заврши и врати тачан одговор)?

Наивно је очекавати позитиван одговор на X Хилбертов проблем због постојања једноставног „полуалгоритма“ којим се „пешачки“ претражује скуп решења. Није тешко смислити програм који би за дати полином $P(x_1, \ldots, x_n)$ у неком поретку набрајао све уређене n-торке целих бројева (почевши, на пример, од $(0, \ldots, 0)$) и за сваку од њих се испитивало да ли јесте решење дате једначине или није. Ако једначина $P(x_1, \ldots, x_n) = 0$ има целобројна решања, тај програм ће се у неком тренутку (колико год он био временски удаљен од почетка извршавања програма) зауставити и вратити нам то решење, Међутим, ако једначина нема решења наш програм се никада неће зауставити.

Улаз: $15x_1^2 - 7x_2^2 = 9$

$$
\begin{aligned}
(0,0): &\quad 15 \cdot 0^2 - 7 \cdot 0^2 \neq 9 \\
(0,1): &\quad 15 \cdot 0^2 - 7 \cdot 1^2 \neq 9 \\
(1,0): &\quad 15 \cdot 1^2 - 7 \cdot 0^2 \neq 9 \\
(1,1): &\quad 15 \cdot 1^2 - 7 \cdot 1^2 \neq 9 \\
(0,2): &\quad 15 \cdot 0^2 - 7 \cdot 2^2 \neq 9 \\
&\vdots \\
(2,1): &\quad 15 \cdot 2^2 - 7 \cdot 1^2 \neq 9 \\
&\vdots
\end{aligned}
$$

Негативан одговор на X Хилбертов проблем последица је чувене Матијасевичеве теореме, која је доказана 1970. године, и која (грубо речено) тврди: *Не постоји поступак о којем Хилберт говори у свом* X *проблему*. Много прецизнију формулацију (и смисао) Матијасевичевог резултата наводимо касније. Једино што овде желимо да истакнемо јесте то да је на почетку наведен интуитивни појам алгоритма немоћан пред проблемима типа: *показати да не постоји алгоритам за решавање неког проблема.*

Аритметика

Термин *коначно* је суштински повезан са интуитивним схватањем речи *алгоритам*. Самим тим, аритметика, у чијем је средишту пажње скуп природних бројева $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, представља математички оквир у којем ћемо увести и проучавати појам алгоритма. У принципу, сва разматрања у вези са коначним скуповима се могу превести на аритметичка разматрања, о чему ће касније бити више речи.

Дедекинд и Пеано су, крајем XIX века, издвојили аксиоме аритметике којима се скуп природних бројева $\mathbb{N}$ описује заједно са константом 0 (нула) и функцијом $s : \mathbb{N} \to \mathbb{N}$ (следбеник):

(P1) $s(n) \neq 0$, за свако n из $\mathbb{N}$; [7]

(P2) Ако је $s(m) = s(n)$, онда је $m = n$, за све m, n из $\mathbb{N}$; [8]

(P3) [**Принцип математичке индукције**] Ако је $S \subseteq \mathbb{N}$ такав да:

(БИ) $0 \in S$,

(ИК) из $n \in S$ следи да $s(n) \in S$, за свако n из $\mathbb{N}$,

онда је $S = \mathbb{N}$. [9]

[7] 0 није следбеник ниједног природног броја; s није на-функција

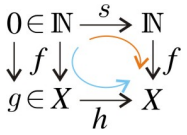
[8] s је 1-1-функција

[9] $\mathbb{N}$ је најмањи (у смислу инклузије) индуктивни скуп.

Једна од најважнијих последица наведених аксиома јесте *Принцип рекурзије*.

Теорема 1. [***Принцип рекурзије***] *Нека је X неки скуп, $g \in X$ и $h : X \to X$. Тада постоји јединствена функција $f : \mathbb{N} \to X$ таква да је*

$$(\text{Rec}) \left| \begin{array}{l} f(0) = g, \\ f(s(n)) = h(f(n)),\ n \in \mathbb{N}. \end{array} \right.$$



$f(0) = g$
$f \circ s(n) = h \circ f(n)$

Када аргумент функције $f : \mathbb{N} \to X$ записујемо у индексу (уместо $f(n)$ пишемо f_n) и уместо $s(n)$ пишемо $n+1$, теорема рекурзије тврди да за изабране $g \in X$ и $h : X \to X$, постоји јединствена функција (тј. јединствени низ) одређен једнакостима:

$$\left| \begin{array}{l} f_0 = g, \\ f_{n+1} = h(f_n),\ n \in \mathbb{N}. \end{array} \right.$$

Доказ. **Егзистенција**.

Скуп $F \subseteq \mathbb{N} \times X$ називаћемо (g,h)-скупом ако су задовољени следећи услови:

1) $(0, g) \in F$ и

2) за све $n \in \mathbb{N}$ и $x \in X$, ако $(n, x) \in F$, онда и $(s(n), h(x)) \in F$.

Очигледно је $\mathbb{N} \times X$ један (g,h)-скуп, па је непразна колекција $\mathcal{F}$ свих (g,h)-скупова. Нека је $f = \bigcap \mathcal{F}$. Тада је $f \subseteq \mathbb{N} \times X$. Доказаћемо да је f заправо тражена функција.[11] Доказ ове чињенице је дугачак само зато што ћемо неколико пута проверавати услове 1) и 2), тј. доказивати да је неки скуп заиста (g,h)-скуп. Да бисмо олакшали читање, ове провере су наведене ситнијим словима и издвојене из остатка доказа.

Докажимо најпре да је $f = \bigcap \mathcal{F}$ један (g,h)-скуп.

[11] Подсећамо на појам функције: скуп f је функција из A у B, у ознаци $f : A \to B$, ако је $f \subseteq A \times B$ и за свако $a \in A$ постоји јединствено $b \in B$ тако да је $(a, b) \in f$. Ако $f : A \to B$, уместо $(a, b) \in f$ пишемо $f(a) = b$.

1) За свако $F \in \mathcal{F}$ важи $(0,g) \in F$, одакле следи да $(0,g) \in \bigcap \mathcal{F} = f$.

2) Претпоставимо да $(n,x) \in f = \bigcap \mathcal{F}$. Тада за свако $F \in \mathcal{F}$, $(n,x) \in F$, па и $(s(n),h(x)) \in F$ (јер F (g,h)-скуп). Дакле, $(s(n),h(x)) \in \bigcap \mathcal{F} = f$.

Даље, доказујемо да је f функција из $\mathbb{N}$ у X, одн. да за свако $k \in \mathbb{N}$ постоји јединствено $y \in X$ тако да $(k,y) \in f$.

(BI) Доказујемо базу индукције ($k = 0$). Како је f један (g,h)-скуп, знамо да $(0,g) \in f$. Претпоставимо да постоји још једно $g' \in X$ такво да $(0,g') \in f$ и $g \neq g'$. Нека је $f' = f \setminus \{(0,g')\}$. Докажимо да је f' један (g,h)-скуп.

1) $(0,g) \in f'$, јер је из f избачен само елемент $(0,g')$ и $g' \neq g$.

2) Претпоставимо да за неке $n \in \mathbb{N}$ и $x \in X$, $(n,x) \in f'$. Будући да тада $(n,x) \in f$ имамо и да $(s(n),h(x)) \in f$. Како је $(s(n),h(x)) \neq (0,g')$, јер је $0 \neq s(n)$, следи да $(s(n),h(x)) \in f'$.

Дакле, $f' \in \mathcal{F}$, па је $f = \bigcap \mathcal{F} \subsetneqq f'$, што је контрадикција.

(IK) Доказујемо индуктивни корак.

IP Претпоставимо да за $k \in \mathbb{N}$ постоји тачно један $y \in X$ такав да је $(k,y) \in f$.

Пошто је f један (g,h)-скуп, имамо да $(s(k),h(y)) \in f$. Претпоставимо да постоји и $y' \in X$ такав да $(s(k),y') \in f$ и $y' \neq h(y)$. Нека је $f' = f \setminus \{(s(k),y')\}$. Докажимо да је f' један (g,h)-скуп.

1) $(0,g) \in f'$, јер је из f избачен само елемент $(s(k),y')$ који је сигурно различит од $(0,g)$, будући да је $0 \neq s(k)$.

2) Претпоставимо да за $n \in \mathbb{N}$ и $x \in X$, $(n,x) \in f'$. Из $(n,x) \in f$ следи $(s(n),h(x)) \in f$. Докажимо да је $(s(n),h(x)) \neq (s(k),y')$. Неједнакост је очигледно тачна ако је $s(n) \neq s(k)$. Ако је $s(n) = s(k)$, онда је и $n = k$, па је $x = y$ (јер је y једини елемент из X такав да $(k,y) \in f$). Према избору елемента y' имамо да је $y' \neq h(y) = h(x)$, одакле следи $(s(n),h(x)) \neq (s(k),y')$.

Дакле, $f = \bigcap \mathcal{F} \subsetneqq f'$, што је контрадикција.

Доказ математичком индукцијом је завршен; доказали смо да је f функција из $\mathbb{N}$ у X. Ова функција задовољава једнакости (Rec), јер је f (g,h)-скуп:

Из услова 1) произлази $f(0) = g$;

Према услову 2), из $f(n) = x$ следи да је $f(s(n)) = h(x)$, тј. $f(s(n)) = h(f(n))$.

Јединственост.

Претпоставимо да функције $f_1, f_2 : \mathbb{N} \to X$ задовољавају једнакости (Rec). Доказаћемо да су оне једнаке, тј. да за свако $n \in \mathbb{N}$ важи $f_1(n) = f_2(n)$. Доказ изводимо математичком индукцијом.

(BI) $f_1(0) = g = f_2(0)$

(IK) Претпоставимо да за неко $n \in \mathbb{N}$ важи $f_1(n) = f_2(n)$. Тада је $f_1(s(n)) = h(f_1(n)) = h(f_2(n)) = f_2(s(n))$. □

Из претходне теореме изводимо и следећу варијанту принципа рекурзије.

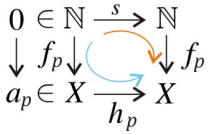
Теорема 2. [***Принцип рекурзије – II***] *Нека* $g : P \to X$ *и* $h : P \times X \to X$. *Тада постоји јединствена функција* $f : P \times \mathbb{N} \to X$ *таква да за свако* $p \in P$:

$$(\text{Rec}) \quad \left| \begin{array}{l} f(p,0) = g(p), \\ f(p,s(n)) = h(p,f(p,n)),\ n \in \mathbb{N}. \end{array} \right.$$

Доказ. Друга варијанта принципа рекурзије блиско је повезана са првом. Заиста, ако $g : P \to X$ и $h : P \times X \to X$, тада за свако $p \in P$, функција g 'бира' један елемент из X – бира $g(p)$ који ћемо означити g_p, а h одређује функцију $h_p : X \to X$, $h_p(x) \overset{\text{def}}{=} h(p,x)$, $x \in X$. Према првој варијанти принципа рекурзије, за свако $p \in P$, постоји јединствена функција $f_p : \mathbb{N} \to X$ таква да:

$$\left| \begin{array}{l} f_p(0) = g_p, \\ f_p(s(n)) = h_p(f_p(n)),\ n \in \mathbb{N}. \end{array} \right.$$

Све функције f_p, $p \in P$, одређују ('подизањем индекса у аргумент') јединствену функцију $f : P \times \mathbb{N} \to X$, $f(p,n) \overset{\text{def}}{=} f_p(n)$ која задовољава услове (Rec). □



Прва варијанта се може сматрати специјалним случајем друге ако изаберемо да S буде синглтон. Нека је $P = \{0\}$. Функцијом $g : \{0\} \to X$ заправо бирамо један елемент из X; нека је $g(0) = g$. Функцију $h : \{0\} \times X \to X$ можемо поистоветити са природно дефинисаном функцијом из X у X: $x \mapsto h(0,x)$, $x \in X$.

Основне рачунске операције уводимо применом друге варијанте теореме рекурзије, узимајући да је $P = X = \mathbb{N}$.

Сабирање. Нека је $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ функција дефинисана са $h(x,y) = s(y)$.[13] Према другој варијанти принципа рекурзије, постоји јединствена функција $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ која задовољава једнакости:

$$\left| \begin{array}{l} +(m,0) = \text{id}_{\mathbb{N}}(m), \\ +(m,s(n)) = h(m,+(m,n)), \end{array} \right. \quad \text{односно} \quad \left| \begin{array}{l} +(m,0) = m, \\ +(m,s(n)) = s(+(m,n)). \end{array} \right.$$

[13] h је композиција следбеника $s : \mathbb{N} \to \mathbb{N}$ и друге пројекције $\Pi^2_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\Pi^2_2(x,y) = y$: $h = s \circ \Pi^2_2$.
$\text{id}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ је идентичка функција, $\text{id}_{\mathbb{N}}(n) = n$.

Ако уместо $+(m,n)$ пишемо $m+n$, претходне једнакости постају:

$$(\text{Rec}+) \quad \left| \begin{array}{l} m+0 = m, \\ m+s(n) = s(m+n). \end{array} \right.$$

Множење. Помоћу константне функције $\mathbf{0} : \mathbb{N} \to \mathbb{N}$, $\mathbf{0}(n) = 0$, и сабирања $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, уводимо множење $\cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ следећим једнакостима:

$$(\text{Rec}\cdot) \quad \left| \begin{array}{l} \cdot(m,0) = \mathbf{0}(m), \\ \cdot(m,s(n)) = +(m,\cdot(m,n)), \end{array} \right. \quad \text{односно} \quad \left| \begin{array}{l} m \cdot 0 = 0, \\ m \cdot s(n) = m + (m \cdot n). \end{array} \right.$$

Степеновање $\exp : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, при чему уместо $\exp(m,n)$ пишемо m^n, дефинишемо следећим једнакостима:

$$\left| \begin{array}{l} m^0 = 1, \\ m^{s(n)} = m \cdot m^n. \end{array} \right.$$

Познати су разлози зашто је израз 0^0 у математици углавном недефинисан. Ипак, по договору узимамо да је $0^0 \overset{\text{def}}{=} 1$, јер ће нам то поједноставити даља разматрања.

Сва позната својства ових операција доказују се математичком индукцијом. Основне бинарне релације скупа $\mathbb{N}$, **уређење** и **дељивост** дефинишемо на следећи начин:[16]

$$m \leqslant n \overset{\text{def}}{\Leftrightarrow} \exists k \in \mathbb{N}\,(m+k=n),\ m \mid n \overset{\text{def}}{\Leftrightarrow} \exists k \in \mathbb{N}\,(m\cdot k=n).$$

[16] Строго уређење је дефинисано са:

$$m < n \overset{\text{def}}{\Leftrightarrow} m \leq n \wedge m \neq n.$$

Према наведеној дефиницији дељивости важи $0 \mid 0$ и $0 \nmid n$, за $n > 0$.

Наредне теореме спадају у базична тврђења за даљи развој аритметике.

Теорема о остатку. *За свe n и $m > 0$ постоје јединствени (количник) q и (остатак) r такви да је $n = qm + r$, $0 \leqslant r < m$.* Ослањајући се на ову теорему, уводимо одговарајуће функције $\mathrm{qt} : \mathbb{N}^2 \to \mathbb{N}$ и $\mathrm{rm} : \mathbb{N}^2 \to \mathbb{N}$, при чему количник и остатак при дељењу нулом одређујемо договором:

$$\mathrm{qt}(m,n) = \begin{cases} \text{количник при дељењу } n \text{ са } m, & m > 0, \\ 0, & m = 0, \end{cases}$$

$$\mathrm{rm}(m,n) = \begin{cases} \text{остатак при дељењу } n \text{ са } m, & m > 0, \\ n, & m = 0. \end{cases}$$

Уместо $\mathrm{qt}(m,n)$ пишемо $\left[\frac{n}{m}\right]$ (уз договор да је $\left[\frac{n}{0}\right] = 0$). Дакле, $\mathrm{rm}(m,n) = n - \left[\frac{n}{m}\right] m$.

Уместо $\mathrm{rm}(m,n)$ пише се и $n \pmod m$.

Теорема о бројевној бази. *Нека је $b > 1$. За свако $a > 0$ постоје јединствени природни бројеви q, k и r такви да је*

$$a = qb^k + r,\ 1 \leqslant q < b, 0 \leqslant r < b^k.$$

Ова теорема омогућава тзв. позициону репрезентацију природног броја цифрама система са базом $b > 1$. До *репрезентације* природног броја n у бази b долазимо на следећи начин: најпре одредимо количник при дељењу са b, а затим сваки добијени количник поново делимо са b све док не добијемо количник једнак нули (а морамо стићи до нуле, јер низ количника опада).

$$\begin{aligned}
&n = q_0 b + r_0,\ 0 \leqslant r_0 < b, \\
&q_0 = q_1 b + r_1,\ 0 \leqslant r_1 < b, (q_1 < q_0) \\
&\qquad n = (q_1 b + r_1)b + r_0 = q_1 b^2 + r_1 b + r_0 \\
&q_1 = q_2 b + r_2,\ 0 \leqslant r_2 < b, (q_2 < q_1) \\
&\qquad n = (q_2 b + r_2)b^2 + r_1 b + r_0 = q_2 b^3 + r_2 b^2 + r_1 b + r_0 \\
&\vdots \\
&q_{k-1} = 0 \cdot b + r_k,\ 1 \leqslant r_k < b, (0 < q_{k-1}) \\
&\qquad n = r_k b^k + \cdots + r_2 b^2 + r_1 b + r_0
\end{aligned}$$

Увођењем симбола за могуће остатке при дељењу са b, поменута репрезентација броја n јесте запис $[r_k \cdots r_1 r_0]_b$. Индекс b изостављамо када је јасно о којој бази је реч.

Познати поступци сабирања и множења природних бројева записаних у датој бројевној бази засновани су на одговарајућим таблицама сабирања и множења цифара. Наводимо таблице сабирања и множења у систему са базом 3:

+	0	1	2
0	0	1	2
1	1	2	**10**
2	2	**10**	**11**

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	**11**

Основна теорема аритметике. *За сваки $n > 1$ постоје јединствени прости бројеви $p_1, p_2, \ldots, p_k$, такви да је $p_1 < p_2 < \cdots < p_k$, и јединствени природни бројеви $\alpha_1, \alpha_2, \ldots, \alpha_k$ тако да је $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$.*

Једнакост $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ се назива **канонска факторизација** броја n. Ако је $p : \mathbb{N} \to \mathbb{N}$ низ простих бројева:

$$p_0 = 2,\ p_1 = 3,\ p_2 = 5,\ p_3 = 7,\ p_4 = 11,\ p_5 = 13,\ p_6 = 17,\ p_7 = 19,\ \ldots$$

за све k и n, означимо $(n)_k$ највећи природан број α такав да $p_k^{\alpha} \mid n$ и $p_k^{\alpha+1} \nmid n$.

На пример, $1960 = 2^3 \cdot 5 \cdot 7^2$, па је $(1960)_0 = 3$, $(1960)_1 = 0$, $(1960)_2 = 1$, $(1960)_3 = 2$, $(1960)_k = 0$ за $k > 3$.

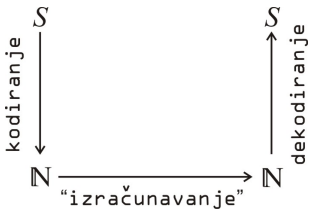
Кодирајуће функције. Скуп је коначан ако постоји нека бијекција између тог скупа и неког природног броја. Једноставна последица аксиома P1 и P2 јесте да сам скуп $\mathbb{N}$ није коначан.[20] Канторово проучавање и класификација бесконачних скупова иницирали су настанак теорије скупова која је значајно утицала на развој савремене математике. На дну лествице бесконачних скупова налазе се *пребројиви скупови*, тј. они скупови чији се елементи могу поређати у низ без понављања елемената. Ређање елемената скупа S у низ без понављања значи дефинисање функције $f : \mathbb{N} \overset{1-1}{\underset{\text{на}}{\to}} S$ којом се набрајају различити елементи скупа S: $f_0, f_1, f_2, \ldots$ Свака бијекција има себи инверзну функцију, што значи да за сваки пребројив скуп S постоји и функција $g : S \overset{1-1}{\underset{\text{на}}{\to}} \mathbb{N}$ коју можемо називати именовање, означавање или *кодирање* елемената скупа S природним бројевима: сваком $s \in S$ на јединствен начин придружујемо природан број који је његово име, ознака, одн. *кôд* при чему различити елементи из S имају различите кодове и сваки природан број је нечији кôд.

[20] Може се доказати да за сваки коначан скуп X и функцију $f : X \to X$ важи:

$$f : X \overset{\text{на}}{\to} X \text{ акко } f : X \overset{1-1}{\to} X.$$

Будући да функција следбеник $s : \mathbb{N} \to \mathbb{N}$ јесте 1-1 функција, а није на-функција, скуп $\mathbb{N}$ није коначан.

Значај кодирања међу првима је уочио Гедел који је пребројиво много ненумеричких објеката именовао природним бројевима и разматрања о полазним објектима преводио у аритметички контекст.



Познато је да се између скупова $\mathbb{N}$ и $\mathbb{N}^k$, за било које $k > 1$, могу успоставити бијекције (обострано-једнозначна пресликавања).

Једно кодирање скупа $\mathbb{N} \times \mathbb{N}$. Једна бијекција између скупова $\mathbb{N}$ и $\mathbb{N}^2$ јесте функција $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$, дата са

$$\langle x_1, x_2 \rangle = 2^{x_1}(2x_2 + 1) - 1.$$

Није тешко показати да је заиста реч о бијекцији. Њена инверзна функција одређена је функцијама $\langle \cdot \rangle_1, \langle \cdot \rangle_2 : \mathbb{N} \to \mathbb{N}$ датим са:

$$\langle n \rangle_1 = (n+1)_0,\ \langle n \rangle_2 = \left[\frac{\frac{n+1}{2^{(n+1)_0}} - 1}{2} \right],\ n \in \mathbb{N}.$$

Прецизније, за све m, n важи:

$$\langle\langle m, n \rangle\rangle_1 = m,\ \langle\langle m, n \rangle\rangle_2 = n,\ \langle \langle n \rangle_1, \langle n \rangle_2 \rangle = n.$$

О кодирању скупова $\mathbb{N}^k$, $k > 2$. Бијективно кодирање уређених парова одређено функцијама $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \overset{1-1}{\underset{\text{на}}{\to}} \mathbb{N}$ и $\langle \cdot \rangle_i : \mathbb{N} \to \mathbb{N}$, $i = 1, 2$, одређује по једно бијективно кодирање сваког од скупова $\mathbb{N}^k$, $k > 2$.

Функцију $g : S \overset{1-1}{\to} \mathbb{N}$ називамо *ефективним кодирањем* или *геделизацијом* уколико су експлицитно задати и следећи ефективни поступци:

- за сваки елемент $s \in S$ може се у коначном броју корака одредити њему придружени број $g(s)$,
- за сваки природан број n може се у коначном броју корака одредити чији је он кôд, тј. може се одредити $s \in S$ за који је $g(s) = n$; овај поступак се назива и *декодирање*.

Појам ефективног поступка за сада прихватамо у интуитивном смислу.

Бијективно кодирање скупа $\mathbb{N}^3$ одређује кодирајућа функција $(x_1, x_2, x_3) \mapsto \langle x_1, \langle x_2, x_3\rangle\rangle$, заједно са декодирајућим функцијама $x \mapsto \langle x\rangle_1$, $x \mapsto \langle\langle x\rangle_2\rangle_1$, $x \mapsto \langle\langle x\rangle_2\rangle_2$.

Бијективно кодирање скупа $\mathbb{N}^4$ одређује кодирајућа функција $(x_1, x_2, x_3, x_4) \mapsto \langle x_1, \langle x_2, \langle x_3, x_4\rangle\rangle\rangle$, заједно са декодирајућим функцијама $x \mapsto \langle x\rangle_1$, $x \mapsto \langle\langle x\rangle_2\rangle_1$, $x \mapsto \langle\langle\langle x\rangle_2\rangle_2\rangle_1$, $x \mapsto \langle\langle\langle x\rangle_2\rangle_2\rangle_2$.

Аналогно претходном, дефинише се по једно кодирање скупа $\mathbb{N}^k$, $k > 4$.

Једно кодирање скупа $\mathbb{N}^{\text{fin}}$ свих коначних низова природних бројева.[22] Подразумева се да скупу $\mathbb{N}^{\text{fin}}$ припада и *празан низ* – низ дужине нула – који ћемо означавати $()$. За сваки непразан коначан низ природних бројева ξ, први члан називаћемо *главом* тог низа и означавати $\text{head}(\xi)$, а остатак (низ добијен изостављањем главе) називаћемо *репом* низа и означавати $\text{tail}(\xi)$. Ове термине и ознаке уводимо да бисмо једноставније описали једну бијекцију $\lceil\cdot\rceil$ између скупа $\mathbb{N}^{\text{fin}}$ и $\mathbb{N}$. Користимо функцију $|\cdot,\cdot| : \mathbb{N}^2 \xrightarrow[\text{на}]{1-1} \mathbb{N}^+$ дату са $|x_1, x_2| = 2^{x_1}(2x_2 + 1)$:

[22] $\mathbb{N}^{\text{fin}} = \bigcup_{k\geqslant 0} \mathbb{N}^k$

На пример,
$\text{head}(1,0,2) = 1$, $\text{tail}(1,0,2) = (0,2)$,
$\text{head}(3,3) = 3$, $\text{tail}(3,3) = (3)$,
$\text{head}(5) = 5$, $\text{tail}(5) = () \ldots$

- $\lceil()\rceil = 0$;
- $\lceil\xi\rceil = |\text{head}(\xi), \lceil\text{tail}(\xi)\rceil|$, за сваки непразан низ ξ.

Број $\lceil\xi\rceil$ називамо кодом низа ξ. Кодови једночланих низова одређени су једнакошћу:

$$\lceil(x_1)\rceil = |\text{head}(x_1), \lceil\text{tail}(x_1)\rceil| = |x_1, \lceil()\rceil| = |x_1, 0| = 2^{x_1};$$

$\lceil(0)\rceil = |0, \lceil()\rceil| = |0,0| = 1$
$\lceil(1)\rceil = |1, \lceil()\rceil| = |1,0| = 2$
$\lceil(2)\rceil = |2, \lceil()\rceil| = |2,0| = 4 \ldots$

кодови двочланих низова:

$$\begin{aligned}\lceil(x_1, x_2)\rceil &= |x_1, \lceil(x_2)\rceil| = |x_1, 2^{x_2}| \\ &= 2^{x_1}\left(2 \cdot 2^{x_2} + 1\right) = 2^{x_1} + 2^{x_1+x_2+1};\end{aligned}$$

$\lceil(0,0)\rceil = |0, \lceil(0)\rceil| = |0,1| = 3$
$\lceil(0,1)\rceil = |0, \lceil(1)\rceil| = |0,2| = 5$
$\lceil(1,0)\rceil = |1, \lceil(0)\rceil| = |1,1| = 6 \ldots$

трочланих:

$$\begin{aligned}\lceil(x_1, x_2, x_3)\rceil &= |x_1, \lceil(x_2, x_3)\rceil| = \left|x_1, 2^{x_2} + 2^{x_2+x_3+1}\right| \\ &= 2^{x_1}\left(2 \cdot \left(2^{x_2} + 2^{x_2+x_3+1}\right) + 1\right) \\ &= 2^{x_1} + 2^{x_1+x_2+1} + 2^{x_1+x_2+x_3+2}.\end{aligned}$$

$\lceil(0,0,0)\rceil = |0, \lceil(0,0)\rceil| = |0,3| = 7$
$\lceil(0,1,0)\rceil = |0, \lceil(1,0)\rceil| = |0,6| = 13 \ldots$

Уопште, једноставно се показује (математичком индукцијом) да за свако $k \geqslant 1$ важи:

$$\begin{aligned}\lceil x_1, \ldots, x_k\rceil &= |x_1, |x_2, \cdots |x_k, 0| \cdots || \\ &= 2^{x_1} + 2^{x_1+x_2+1} + \cdots + 2^{x_1+x_2+\cdots+x_k+k-1}.\end{aligned}$$

На основу ове једнакости једноставно налазимо бинарну репрезентацију кода било ког непразног коначног низа природних бројева:

$$\lceil x_1, \ldots, x_k\rceil = [1\underbrace{0\ldots0}_{x_k \text{ нула}}1\ldots1\underbrace{0\ldots0}_{x_1 \text{ нула}}]_2.$$

Пример 1. Одредимо кодове неколико непразних коначних низова:
$\lceil 3,2,4\rceil = [1\underbrace{0000}_{4}1\underbrace{00}_{2}1\underbrace{000}_{3}]_2 = 2^3+2^6+2^{11} = 2120,$
$\lceil 1,2,0,1\rceil = [10110010]_2 = 2^1+2^4+2^5+2^7 = 178,$
$\lceil 0,0,1,0,0,0\rceil = [1110111]_2 = 2^0+2^1+2^2+2^4+2^5+2^6 = 119\ldots$

Сваки природан број x јесте кôд неког коначног низа природних бројева. Нула је кôд празног низа. Број $x > 0$ је кôд неког непразног коначног низа који једноставно одређујемо на основу бинарне репрезентације броја x. Одредимо коначан низ чији је кôд 372. Бинарна репрезентација броја 372 јесте $[101110100]_2$, одакле непосредно уочавамо да је $372 = \lceil 2,1,0,0,1\rceil$.

Једно кодирање скупа $\mathcal{P}_{\text{fin}}(\mathbb{N})$ свих коначних подскупова од $\mathbb{N}$. Нека је $v : \mathcal{P}_{\text{fin}}(\mathbb{N}) \to \mathbb{N}$ функција дефинисана са:

$$v(\emptyset) = 0,\ v(\{n_1,\ldots,n_k\}) = 2^{n_1}+\cdots+2^{n_k}.$$

Уместо доказа да је v бијекција, истичемо само ефективност декодирања: да бисмо одредили коначан скуп чији је кôд n, довољно је n приказати у бинарном запису и из тог записа прочитати елементе траженог скупа.

Број $372 = 2^2+2^4+2^5+2^6+2^8$ је кôд скупа $\{2,4,5,6,8\}$.

Дијагонализација

Кодирање скупа $\mathbb{N}^{\mathbb{N}}$ није могуће. Скуп $\mathbb{N}$ нема довољно елемената за кодирање свих низова природних бројева, тј. свих функција из $\mathbb{N}$ у $\mathbb{N}$. Изостављајући неке формалне детаље, истичемо само основне идеје доказа ове чињенице. Када год низове природних бројева поређамо у низ:

$$\begin{aligned}
x_0 &: n_0^0,\ n_1^0,\ n_2^0,\ n_3^0,\ n_4^0,\ \cdots\\
x_1 &: n_0^1,\ n_1^1,\ n_2^1,\ n_3^1,\ n_4^1,\ \cdots\\
x_2 &: n_0^2,\ n_1^2,\ n_2^2,\ n_3^2,\ n_4^2,\ \cdots\\
x_3 &: n_0^3,\ n_1^3,\ n_2^3,\ n_3^3,\ n_4^3,\ \cdots\\
&\vdots
\end{aligned}$$

увек је могуће дефинисати низ природних бројева који се не налази на тој листи: нека је (m_k) низ природних бројева такав да за свако k важи $m_k \neq n_k^k$. Низ (m_k) не налази се на листи, јер је различит од сваког низа листе.

Метод који смо овде применили назива се *дијагонализација* и биће један од кључних метода које ћемо користити у наставку.

Кодирање скупа $\mathcal{P}(\mathbb{N})$ није могуће. Скуп $\mathcal{P}(\mathbb{N})$, свих (и коначних и бесконачних) подскупова од $\mathbb{N}$, можемо идентификовати са скупом $2^{\mathbb{N}}$ свих бинарних низова: сваки $A \subseteq \mathbb{N}$ одређује јединствену функцију $\chi_A : \mathbb{N} \to \{0,1\}$,

$$\chi_A(n) = \begin{cases} 1, & n \in A,\\ 0, & n \notin A, \end{cases}$$

а сваки бинарни низ $f : \mathbb{N} \to \{0,1\}$ одређује јединствени подскуп $A_f = \{n \in \mathbb{N} \mid f(n) = 1\}$ скупа $\mathbb{N}$.

Дијагонализацијом једноставно доказујемо да није могуће све бинарне низове поређати у низ.

Алфабет, реч, језик

Користећи уобичајени позициони систем приказивања природних бројева у некој бази $b > 1$, ми заправо сваком природном броју придружујемо једну реч (коначан низ симбола) алфабета који садржи тачно b симбола – по један симбол за сваки од бројева $0, 1, \ldots, b-1$. При томе, не употребљавамо речи које садрже више од једне цифре, а почињу цифром 0. Када не бисмо искључили ове речи, природни број не би имао јединствену репрезентацију.[28] Ову неједнозначност изазива употреба симбола 0 у репрезентацијама бројева већих од нуле, и можемо је избећи ако незнатно изменимо начин репрезентације. За приказивање бројева већих од нуле, за дату базу $b > 1$ можемо увести цифре (симболе) за бројеве $1, \ldots, b$ и ослонити се на директну последицу леме о остатку: *За сваки природан број $n > 0$ постоје јединствени q и r такви да је $n = qb + r$, $1 \leqslant r \leqslant b$.* Користећи ово тврђење закључујемо да за сваки природан број $n > 0$ постоје јединствени k, $r_k, \ldots, r_1, r_0$ такви да је

$$n = r_k b^k + \cdots + r_1 b^1 + r_0,\ 1 \leqslant r_k, \ldots, r_1, r_0 \leqslant b.$$

Запис $[r_k \ldots r_1 r_0]_b$ називаћемо репрезентацијом броја n у *модификованом* систему базе b. На овај начин, природно се успоставља бијекција између скупа $\mathbb{N}$ и свих речи алфабета са b симбола, при чему нули одговара празна реч.

[28] Записи

$$12 = 1 \cdot 10 + 2$$
$$012 = 0 \cdot 10^2 + 1 \cdot 10 + 2$$
$$0012 = 0 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10 + 2$$
$$\vdots$$

одговарају истом природном броју.

Пример 2. Изаберимо за цифре модификованог декадног система следеће симболе: 1, 2, 3, 4, 5, 6, 7, 8, 9, X. Није тешко запис броја у уобичајеном декадном систему превести у модификован, и обратно. Броју 3001 (записаном у уобичајеном декадном систему), у модификованом декадном систему одговара запис 29X1:

$$\begin{aligned} 3001 &= 300 \cdot 10 + \mathbf{1}, \\ 300 &= 29 \cdot 10 + \mathbf{X}, \\ 29 &= 2 \cdot 10 + \mathbf{9}, \\ 2 &= 0 \cdot 10 + \mathbf{2}. \end{aligned}$$

Запису 2X3X одговара следећи запис у уобичајеном декадном систему:

$$2X3X = 2 \cdot 10^3 + X \cdot 10^2 + 3 \cdot 10 + X = 2000 + 1000 + 30 + 10 = 3040.$$

Познате поступке сабирања и множења без проблема прилагођавамо модификованом систему користећи одговарајуће таблице.

0										
1	2	3	4	5	6	7	8	9	X	
11	12	13	14	15	16	17	18	19	1X	
21	22	23	24	25	26	27	28	29	2X	
⋮										
91	92	93	94	95	96	97	98	99	9X	
X1	X2	X3	X4	X5	X6	X7	X8	X9	XX	
111	112	113	114	115	116	117	118	119	11X	
⋮										

Таблице сабирања и множења у модификованом систему базе 3:

+	1	2	3
1	2	3	**11**
2	3	**11**	**12**
3	**11**	**12**	**13**

·	1	2	3
1	1	2	3
2	2	**11**	**13**
3	3	**13**	**23**

У модификоване позиционе системе можемо укључити и систем са базом 1, у коме се природни бројеви приказују само једним симболом, на пример знаком 1:

$$\varepsilon,\ 1,\ 11,\ 111,\ 1111,\ \cdots$$

Репрезентација природних бројева помоћу једног симбола представља „најпримитивнији“ начин записивања бројева, ако узмемо у обзир да су, у зачецима људске цивилизације, количине бележене писањем *рецки*. Поред тога, записивање бројева једним симболом сагласно је и са репрезентацијама бројева на које нас директно наводи језик Пеанове аксиоматике:

$$0,\ s(0),\ s(s(0)),\ s(s(s(0))),\ \ldots$$

Претходна прича о позиционим системима заправо мотивише и илуструје појмове: *алфабет*, *реч* и *језик*. Под *алфабетом* Σ подразумевамо било који непразан коначан скуп чије елементе називамо *симболима*. Сваки алфабет користимо да бисмо написали неки текст. Наводимо неколико познатих алфабета:

- $\Sigma_{\mathrm{U}} = \{1\}$ – *унарни алфабет*, који садржи само један симбол;
- $\Sigma_2 = \{0,1\}$ – *бинарни алфабет*, веома важан за рачунарство;
- $\Sigma_{\mathrm{lat}} = \{a,b,c,\ldots,z\}$ – мала слова латиница;
- $\Sigma_{\mathrm{keyboard}} = \{\mathtt{A},\mathtt{a},\mathtt{B},\mathtt{b},\ldots,\mathtt{Z},\mathtt{z},\sqcup,>,<,(,),\ldots,!\}$ је алфабет свих симбола тастатуре, при чему $\sqcup$ означава бланко знак.

Рачунари раде са текстовима које можемо посматрати као низове симбола над неким задатим алфабетом (који чине сви знаци који се могу откуцати на тастатури). Програми су текстови над алфабетом тастатуре, улази и излази су такође неки текстови над истим алфабетом, при чему програм трансформише улазни текст у излазни.

Ако је Σ алфабет, за $m \geqslant 2$, скуп $\Sigma^m = \underbrace{\Sigma \times \cdots \times \Sigma}_{m \text{ пута}}$ свих m-торки симбола називамо скупом свих *речи над* Σ *дужине* m, и уместо $(a_1,\ldots,a_m)$ пишемо $a_1 \cdots a_m$. Скуп Σ^1, тј. скуп свих речи над Σ дужине 1, идентификујемо са скупом Σ. Скуп Σ^0 је једночлани скуп чији ћемо елемент означавати ε и називати га *празна реч*. Скуп свих речи над алфабетом Σ означавамо Σ^*, а скуп свих непразних речи Σ^+:

$$\Sigma^+ = \bigcup_{m\geqslant 1} \Sigma^m,\ \Sigma^* = \bigcup_{m\geqslant 0} \Sigma^m = \Sigma^+ \cup \{\varepsilon\}.$$

$\Sigma_2^0 = \{\varepsilon\}, \Sigma_2^1 = \{0,1\},$
$\Sigma_2^2 = \{00,01,10,11\},$
$\vdots$
$\Sigma_2^* = \{\varepsilon,0,1,00,01,10,11,000,001,\ldots\}$
$\Sigma_2^+ = \{0,1,00,01,10,11,000,001,\ldots\}$

Ако $w \in \Sigma^*$, тада јединствени m такав да $w \in \Sigma^m$ називамо дужином речи w и пишемо $|w| = m$. Приметимо да је $|\Sigma^m| = |\Sigma|^m$.

На скупу Σ^* дефинишемо бинарну операцију **дописивања** (конкатенације): ако је $u = a_1 \cdots a_k$, $v = b_1 \cdots b_\ell \in \Sigma^*$, онда је uv реч $a_1 \ldots a_k b_1 \ldots b_\ell$. Ова операција је асоцијативна, $(uv)w = u(vw)$, и ε је неутрални елемент, $w\varepsilon = \varepsilon w = w$, па је над Σ^* дефинисан један моноид, такозвани *слободни моноид*. Као и обично, за $w \in \Sigma^*$ и $n \in \mathbb{N}$ дефинишемо w^n са: $w^0 = \varepsilon$, $w^{n+1} = w^n w$. Ако $w \in \Sigma^*$, *подреч* речи w је свака реч $u \in \Sigma^*$, таква да је $w = v_1 u v_2$, за неке $v_1, v_2 \in \Sigma^*$. Ако је $w = uv$, онда се каже да је u *префикс* речи w, као и да је v *суфикс* речи w. Приметимо да ако Σ има бар два симбола, онда операција дописивања скупа Σ^* *није комутативна*.

Нека је $\Sigma = \{s_1,\ldots,s_m\}$ алфабет чији су симболи линеарно уређени: $s_1 < s_2 < \ldots < s_m$. На скупу Σ^* уводимо тзв. **строги лексикографски поредак** $<_{\mathrm{lex}}$ на следећи начин:

$$u <_{\mathrm{lex}} v \Leftrightarrow\ (|u| < |v|) \vee$$
$$\vee\, (|u| = |v| \wedge u = ws_i x \wedge v = ws_j y \text{ за неке } w,x,y \in \Sigma^* \text{ и } s_i < s_j).$$

Наравно, $u \leqslant_{\text{lex}} v \Leftrightarrow u <_{\text{lex}} v \vee u = v$. Није тешко приметити да је $(\Sigma^*, \leqslant_{\text{lex}})$ добро уређење, и да свака реч из Σ^* има непосредног следбеника у односу на $\leqslant_{\text{lex}}$. Специјално, ако је $\Sigma = \{0,1\}$ и $0 < 1$, онда је:
$\varepsilon <_{\text{lex}} 0 <_{\text{lex}} 1 <_{\text{lex}} 00 <_{\text{lex}} 01 <_{\text{lex}} 10 <_{\text{lex}} 11 <_{\text{lex}} 000 <_{\text{lex}} 001 <_{\text{lex}} 010 <_{\text{lex}} 011 <_{\text{lex}} 100 <_{\text{lex}} \cdots$

Дефиниција 1. ***Језик*** *над алфабетом* Σ *је било који подскуп од* Σ^*.

ПРИМЕР 3. Тривијални језици над било којим алфабетом Σ јесу: $\emptyset$ (празан језик), $\{\varepsilon\}$ (језик који садржи само празну реч) и Σ^* (скуп свих речи над Σ). Наводимо неколико нетривијалних језика над $\{a,b\}$:

$L_1 = \{\varepsilon, ab, bab\}$;

$L_2 = \{a^p \mid p \text{ је прост број}\} = \{aa, aaa, aaaaa, aaaaaaa, \ldots\}$;

$L_3 = \{a^i b^{i+j} a^j \mid i,j \geqslant 1\}$;

$L_4 = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$, при чему, за $s \in \{a,b\}$, $|w|_s$ означава број појављивања симбола s у речи w.

Како су језици скупови, на њих примењујемо стандардне скуповне операције. Нека су L_1 и L_2 језици над Σ.

- *Унија* језика L_1 и L_2 јесте језик

$$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ или } w \in L_2\}.$$

- *Пресек* језика L_1 и L_2 јесте језик

$$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ и } w \in L_2\}.$$

- *Комплемент* језика L_1 јесте језик

$$L_1^{\complement} = \Sigma^* \setminus L_1 = \{w \in \Sigma^* \mid w \notin L_1\}.$$

Поред наведених операција, важна операција међу језицима L_1 и L_2 истог алфабета Σ јесте *надовезивање језика*:

$$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}.$$

Користећи надовезивање језика, за било који језик L уводимо следеће језике: $L^0 = \{\varepsilon\}$, $L^{i+1} = L^i L$, $i \in \mathbb{N}$. Језик $L^* = \bigcup\limits_{i \geqslant 0} L^i$ назива се *Клинијева звездица* језика L.

ТЈУРИНГОВЕ МАШИНЕ

Замислимо траку, издељену на ћелије (поља), која је неограничена са обе стране. У сваку ћелију траке може бити уписан само један симбол неког унапред изабраног алфабета $\Gamma = \{s_1, \ldots, s_m\}$ међу којима се налази и један специјалан симбол, тзв. *бланко знак* „$\sqcup$“ (тј. $\sqcup = s_i$, за неко i). Поред тога, замислимо и да постоји механизам, назовимо га *глава*, који може да чита садржај само једне ћелије и да извршава једну од инструкција **Тјуринговог програма**, тј. једну од инструкција коначног низа инструкција означених q_0, q_1, …, q_k, при чему је свака инструкција q_i следећег облика, где $D_1, \ldots, D_m$ узимају вредности R или L или P:

$$q_i \left| \begin{array}{l} s_1 s_1' D_1 q_{j_1} \\ \quad\vdots \\ s_m s_m' D_m q_{j_m} \end{array} \right.$$

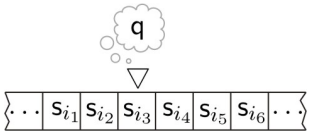
ЗНАЧЕЊЕ: Ако је s_ℓ симбол који читаш, обриши га и упиши s_ℓ', помери се на лево (ако је D_ℓ једнако L), одн. десно (ако је D_ℓ једнако R) суседно поље или остани на истом пољу (ако је D_ℓ једнако P) и пређи на извршавање инструкције q_{j_ℓ} ($s_\ell, s_\ell' \in \Gamma, 1 \leqslant \ell \leqslant m, 0 \leqslant j_1, \ldots, j_m \leqslant k$).

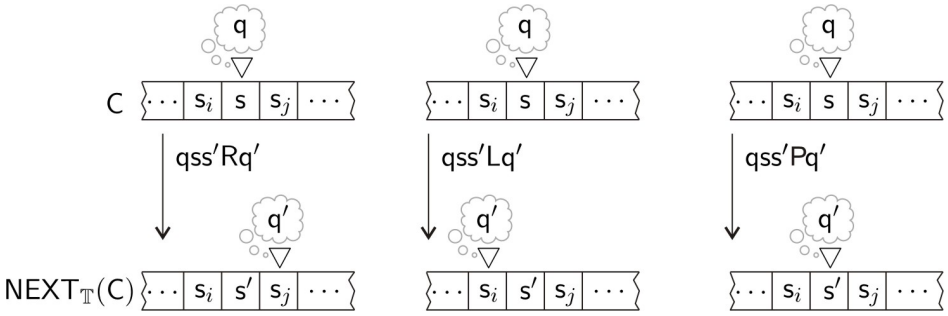
или

q_i STOP ЗНАЧЕЊЕ: Прекини са радом.

Инструкције првог облика можемо записати и као низ уређених петорки: $q_i s_1 s_1' D_1 q_{j_1}$, …, $q_i s_m s_m' D_m q_{j_m}$. Ознаку инструкције називамо и **стањем**, при чему ознаку оне којом треба започети извршавање програма (најчешће q_0) издвајамо као **почетно стање**. Ознаке свих STOP инструкција називамо **завршним стањима**. Када кажемо да се глава, читајући садржај неке ћелије, налази у стању q_i, то значи да треба извршити инструкцију q_i.

Када је познат садржај сваке ћелије траке, положај главе и стање у којем се она налази кажемо да је позната **конфигурација** траке. **Завршне конфигурације** су оне конфигурације у којима је глава у неком завршном стању. Уколико конфигурација C није завршна, онда је Тјуринговим програмом потпуно одређена наредна конфигурација $\mathrm{NEXT}_{\mathbb{T}}(C)$.

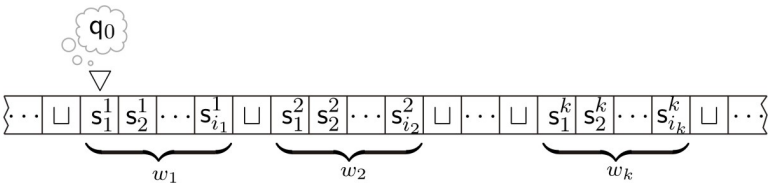




Извршавање Тјурингових програма посматраћемо у ситуацијама када је на почетку на траци уписано коначно много речи унапред одређеног **улазног алфабета** $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$ (Σ не садржи бланко знак!), при чему су:

- симболи сваке речи уписани слева надесно у суседне ћелије траке (у једну ћелију може бити уписан само један симбол),
- речи су раздвојене једним бланко знаком и
- у свим осталим ћелијама уписани бланко знаци.

Овако одређен почетни садржај траке, уз договор да глава у почетном стању скенира прву ћелију слева од које почиње унос улаза називамо **почетном конфигурацијом** траке. Ако на улазу треба унети k речи алфабета Σ, $w_1 = \mathsf{s}_1^1\mathsf{s}_2^1\cdots\mathsf{s}_{i_1}^1$, $w_2 = \mathsf{s}_1^2\mathsf{s}_2^2\cdots\mathsf{s}_{i_2}^2$, $\ldots$, $w_k = \mathsf{s}_1^k\mathsf{s}_2^k\cdots\mathsf{s}_{i_k}^k$, одговарајућа почетна конфигурација је приказана на слици испод.



У случају да је w_1 празна реч, глава у почетном стању чита први слева бланко знак испред речи w_2.

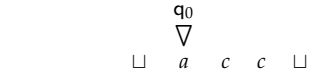
Када окупимо у целину све што је релевантно за извршавање Тјурингових програма, добијамо формални опис Тјурингове машине. Наиме, **Тјурингова машина** (или краће ТМ) јесте уређена седморка $\mathbb{T} = (Q, \mathsf{q}_0, F, \Gamma, \sqcup, \Sigma, \tau)$ при чему је:

- Q коначан скуп стања, тј. скуп свих инструкција програма;
- $\mathsf{q}_0 \in Q$ је почетно стање, тј. прва инструкција којом се започиње извршавање програма,
- $F \subseteq Q$ скуп завршних стања, тј. скуп STOP-инструкција,
- Γ је алфабет траке који садржи бланко знак $\sqcup$, тј. скуп свих симбола који могу бити уписивани у ћелије траке,
- $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$ улазни алфабет, и
- τ је заправо Тјурингов програм; идентификујемо га са тзв. функцијом транзиције $\tau : (Q \setminus F) \times \Gamma \to \Gamma \times \{\mathsf{R}, \mathsf{L}, \mathsf{P}\} \times Q$, којом се описују инструкције које нису STOP-инструкције: ако се уређена петорка $\mathsf{qss'Dq'}$ појављује као део инструкције q, онда је $\tau(\mathsf{q}, \mathsf{s}) = (\mathsf{s}', \mathsf{D}, \mathsf{q}')$.

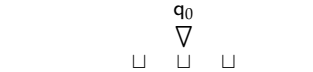
Најзначајнији део Тјурингове машине јесте њен програм (описан као функција транзиције). Због једноставнијег излагања, у наставку ћемо Тјурингове машине углавном задавати прецизирајући улазни алфабет, број речи улазног алфабета које очекујемо на улазу и наводећи функцију транзиције, тј. одговарајући програм (изостављајући STOP инструкције).

Ако је $\Sigma = \{a, b, c\}$ улазни алфабет, наводимо почетне конфигурације за неколико различитих улаза.

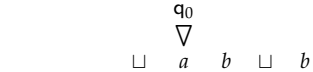
улаз *acc*:



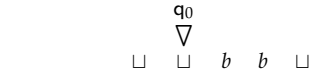
улаз ε:



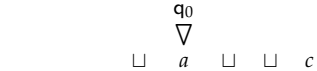
улаз (ab, b):



улаз (ε, bb):



улаз (a, ε, c):



$$\mathsf{q}_i \;\Big|\; \begin{array}{l} \mathsf{s}_1\mathsf{s}_1'\mathsf{D}_1\mathsf{q}_{j_1} \\ \quad\vdots \\ \mathsf{s}_m\mathsf{s}_m'\mathsf{D}_m\mathsf{q}_{j_m} \end{array}$$

$\tau(\mathsf{q}_i, \mathsf{s}_1) = (\mathsf{s}_1', \mathsf{D}_1, \mathsf{q}_{j_1})$

$\vdots$

$\tau(\mathsf{q}_i, \mathsf{s}_m) = (\mathsf{s}_m', \mathsf{D}_m, \mathsf{q}_{j_m})$

Пример 4. Тјурингов програм[38] задат следећим инструкцијама:

$$\mathsf{q}_0 \sqcup \sqcup \mathsf{L}\mathsf{q}_2 \quad \mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1 \quad \mathsf{q}_2\,\mathsf{STOP}$$
$$\mathsf{q}_0 1 \sqcup \mathsf{R}\mathsf{q}_1 \quad \mathsf{q}_1 1 \sqcup \mathsf{R}\mathsf{q}_0$$

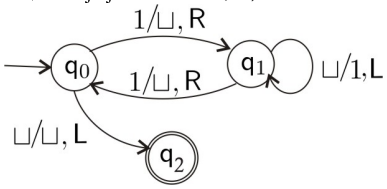
уз договор да је $\Sigma = \{1\}$ улазни алфабет, одређује ТМ

$$\mathbb{T} = (\{\mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2\}, \mathsf{q}_0, \{\mathsf{q}_2\}, \{\sqcup, 1\}, \sqcup, \{1\}, \tau),$$

где је $\tau : (Q \setminus F) \times \Gamma \to \Gamma \times \{\mathsf{R}, \mathsf{L}\} \times Q$ задато табелом:

τ	$\sqcup$	1
q_0	$(\sqcup, \mathsf{L}, \mathsf{q}_2)$	$(\sqcup, \mathsf{R}, \mathsf{q}_1)$
q_1	$(1, \mathsf{L}, \mathsf{q}_1)$	$(\sqcup, \mathsf{R}, \mathsf{q}_0)$

[38] Сваки Тјурингов програм је погодно приказати цртежом који подсећа на граф. Чворови графа су означени стањима; на почетно стање указује стрелица из спољашњости; завршна стања су истакнута концентричним круговима; ивице између два чвора означене су у складу са наредбама програма: за сваку наредбу $\mathsf{q}\mathsf{s}\mathsf{s}'\mathsf{D}\mathsf{q}'$ чворови q и q' су спојени ивицом која је означена $\mathsf{s}/\mathsf{s}', \mathsf{D}$.



Машина $\mathbb{T}$, за сваки улаз облика 1^n, $n \geqslant 0$, генерише један низ конфигурација који може бити коначан или бесконачан. Посматрајмо извршавање наведеног програма за три различите улазне речи: ε, 1 и 11.

q_0 ∇ $\cdots \sqcup \sqcup \sqcup \cdots$ $\downarrow \mathsf{q}_0 \sqcup \sqcup \mathsf{L}\mathsf{q}_2$ q_2 ∇ $\cdots \sqcup \sqcup \sqcup \cdots$ $\mathsf{q}_2\,\mathsf{STOP}$	q_0 ∇ $\cdots \sqcup \sqcup \sqcup 1 \sqcup \cdots$ $\downarrow \mathsf{q}_0 1 \sqcup \mathsf{R}\mathsf{q}_1$ q_1 ∇ $\cdots \sqcup \sqcup \sqcup \sqcup \sqcup \cdots$ $\downarrow \mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1$ q_1 ∇ $\cdots \sqcup \sqcup \sqcup \sqcup 1 \cdots$ $\downarrow \mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1$ q_1 ∇ $\cdots \sqcup \sqcup \sqcup 1 1 \cdots$ $\downarrow \mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1$ q_1 ∇ $\cdots \sqcup \sqcup 1 1 1 \cdots$ $\downarrow \mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1$ $\vdots$	q_0 ∇ $\cdots \sqcup 1 1 \sqcup \cdots$ $\downarrow \mathsf{q}_0 1 \sqcup \mathsf{R}\mathsf{q}_1$ q_1 ∇ $\cdots \sqcup \sqcup 1 \sqcup \cdots$ $\downarrow \mathsf{q}_1 1 \sqcup \mathsf{R}\mathsf{q}_0$ q_0 ∇ $\cdots \sqcup \sqcup \sqcup \sqcup \cdots$ $\downarrow \mathsf{q}_0 \sqcup \sqcup \mathsf{L}\mathsf{q}_2$ q_2 ∇ $\cdots \sqcup \sqcup \sqcup \sqcup \cdots$ $\mathsf{q}_2\,\mathsf{STOP}$

Извршавање програма (израчунавање машине) $\mathbb{T}$ се зауставља за улазе ε и 11, али се не зауставља за улаз 1.

Свака Тјурингова машина $\mathbb{T} = (Q, \mathsf{q}_0, F, \Gamma, \sqcup, \Sigma, \tau)$ за било који улаз $\overline{w} = (w_1, \ldots, w_k) \in \Sigma^{*k}$, тј. одговарајућу почетну конфигурацију $\mathsf{C}_{\overline{w}}$, генерише један низ конфигурација:

$$\Sigma^{*k} \stackrel{\text{def}}{=} \underbrace{\Sigma^* \times \cdots \times \Sigma^*}_{k \text{ пута}}$$

$$\mathsf{C}_{\overline{w}} \to \mathsf{NEXT}_{\mathbb{T}}(\mathsf{C}_{\overline{w}}) \to \mathsf{NEXT}_{\mathbb{T}}(\mathsf{NEXT}_{\mathbb{T}}(\mathsf{C}_{\overline{w}})) \to \cdots$$

који је:

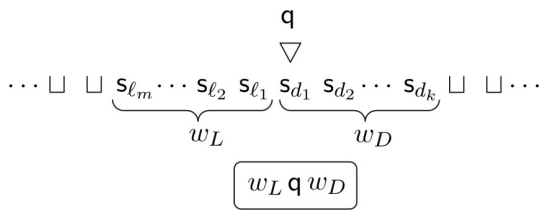
- *коначан* ако се у низу појави завршна конфигурација; тада пишемо $\mathbb{T}(\overline{w})\downarrow$ и кажемо $\mathbb{T}$ *се зауставља (конвергира) за улаз* $\overline{w}$; или
- *бесконачан* ако се у низу никада не појављује завршна конфигурација; тада пишемо $\mathbb{T}(\overline{w})\uparrow$ и кажемо $\mathbb{T}$ *се не зауставља (дивергира) за улаз* $\overline{w}$.

Ако $\mathbb{T}(\overline{w})\downarrow$, коначан низ конфигурација $\mathsf{C}_{\overline{w}}, \mathsf{C}_1, \ldots, \mathsf{C}_d$, где је:

- $\mathsf{C}_{\overline{w}}$ почетна конфигурација одређена улазом $\overline{w}$,
- $\mathsf{C}_{i+1} = \mathsf{NEXT}_{\mathbb{P}}(\mathsf{C}_i)$, $1 \leqslant i < d$; пар суседних конфигурација називамо *рачунским кораком*; и
- C_d завршна конфигурација,

називамо **израчунавањем** машине $\mathbb{T}$ за улаз $\overline{w}$ и пишемо $\mathsf{C}_{\overline{w}} \to_{\mathbb{T}}^{*} \mathsf{C}_d$. Ако $\mathbb{T}(\overline{w})\uparrow$, тј. низ конфигурација је бесконачан, израчунавање машине $\mathbb{T}$ се не зауставља за улаз $\overline{w}$ и пишемо $\mathsf{C}_{\overline{w}} \to_{\mathbb{T}}^{*} \infty$.

Будући да у почетној конфигурацији само коначно много ћелија траке садржи неки симбол различит од бланко знака, то ће важити и за све наредне конфигурације израчунавања било коју ТМ $\mathbb{T}$. Зато ћемо сваку конфигурацију означити као реч језика $\Gamma^* Q \Gamma^+$, тј. у облику $w_L \mathsf{q} w_D$, $w_L \in \Gamma^*$, $w_D \in \Gamma^+$, при чему је:



- w_L најкраћа реч која се на траци може прочитати слева надесно до ћелије коју чита глава, не укључујући и ту ћелију, и која обухвата све симболе поменутих ћелија различите од $\sqcup$; уколико лево од ћелије која се чита нема симбола различитих од $\sqcup$, онда је $w_L = \varepsilon$.
- w_D најкраћа реч која се на траци може прочитати слева надесно од ћелије коју чита глава, укључујући и ту ћелију, и која обухвата све симболе поменутих ћелија различите од $\sqcup$; уколико десно од ћелије која се чита нема симбола различитих од $\sqcup$, онда је $w_D = \sqcup$.

Узимајући о обзир овакво означавање конфигурација, приликом извршавања произвољне ТМ $\mathbb{T} = (Q, \mathsf{q}_0, F, \Gamma, \sqcup, \Sigma, \tau)$, кључну улогу има функција $\mathsf{NEXT}_{\mathbb{T}} : \Gamma^*(Q \setminus F)\Gamma^+ \to \Gamma^* Q \Gamma^+$ коју прецизно дефинишемо на следећи начин:

$$\mathsf{NEXT}_{\mathbb{T}}(w_L \mathsf{q} \mathsf{s} w_D) = \begin{cases} w_L \mathsf{s}' \mathsf{q}' w_D, & \text{ако је } \tau(\mathsf{q},\mathsf{s}) = (\mathsf{s}', \mathsf{R}, \mathsf{q}') \text{ и } w_D \neq \varepsilon, \\ w_L \mathsf{s}' \mathsf{q}' \sqcup, & \text{ако је } \tau(\mathsf{q},\mathsf{s}) = (\mathsf{s}', \mathsf{R}, \mathsf{q}') \text{ и } w_D = \varepsilon, \\ w \mathsf{q}' \mathsf{s}_i \mathsf{s}' w_D, & \text{ако је } \tau(\mathsf{q},\mathsf{s}) = (\mathsf{s}', \mathsf{L}, \mathsf{q}') \text{ и } w_L = w \mathsf{s}_i, \\ \mathsf{q}' \sqcup \mathsf{s}' w_D, & \text{ако је } \tau(\mathsf{q},\mathsf{s}) = (\mathsf{s}', \mathsf{L}, \mathsf{q}') \text{ и } w_L = \varepsilon, \\ w_L \mathsf{q}' \mathsf{s}' w_D, & \text{ако је } \tau(\mathsf{q},\mathsf{s}) = (\mathsf{s}', \mathsf{P}, \mathsf{q}'). \end{cases}$$

Пример 5. Користећи уведено означавање конфигурација, израчунавања ТМ[41] $\mathbb{T}$ из примера 4 можемо приказати и на следећи начин:
$q_0\sqcup \to q_2\sqcup$; дакле, $q_0\sqcup \to^* q_2\sqcup$ и $\mathbb{T}(\varepsilon)\downarrow$;
$q_01 \to q_1\sqcup \to q_1\sqcup 1 \to q_1\sqcup 11 \to q_1\sqcup 111 \to \cdots$; видимо $\mathbb{T}(1)\uparrow$;
$q_011 \to q_11 \to q_0\sqcup \to q_2\sqcup$; дакле, $q_011 \to^* q_2\sqcup$ и $\mathbb{T}(11)\downarrow$.

Математичком индукцијом се може доказати да:

$$q_01^n \to^* \begin{cases} q_2\sqcup, & \text{ако је } n \text{ паран;} \\ \infty, & \text{ако је } n \text{ непаран.} \end{cases}$$

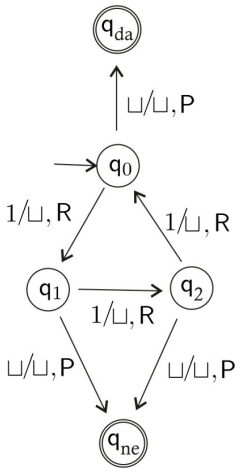
Пример 6. Посматрајмо Тјурингову машину

$$\mathbb{T} = (\{q_0, q_1, q_1, q_{da}, q_{ne}\}, q_0, \{q_{da}, q_{ne}\}, \{1\}, \sqcup, \{0, 1, \sqcup\}, \tau)$$

при чему је τ одређено следећим наредбама:

$$\begin{array}{lll} q_0\sqcup\sqcup \mathsf{P} q_{da} & q_1\sqcup\sqcup \mathsf{P} q_{ne} & q_2\sqcup\sqcup \mathsf{P} q_{ne} \\ q_01\sqcup \mathsf{R} q_1 & q_11\sqcup \mathsf{R} q_2 & q_21\sqcup \mathsf{R} q_0 \end{array}$$

за улазни алфабет $\Sigma = \{1\}$.



Посматраћемо израчунавања машине $\mathbb{T}$ искључиво за улазе које чини једна реч алфабета $\{1\}$. За улаз $1^0 = \varepsilon$, израчунавање ће се завршити после једног рачунског корака и завршна конфигурација ће бити $q_{da}\sqcup$. За улаз 1^n, $n \geqslant 1$, глава ће учитавати симбол по симбол слева надесно бришући садржај ћелија (тј. уписујући $\sqcup$), и када стигне до бланко симбола у стању q_0 отићи ће у стање q_{da} и зауставити се, а ако стигне до бланко симбола у стању q_1 или q_2 отићи у стање q_{ne} и зауставити се. Стање у којем се машина зауставља можемо тумачити као одговор на питање „да ли је број n дељив са 3 или није?“.

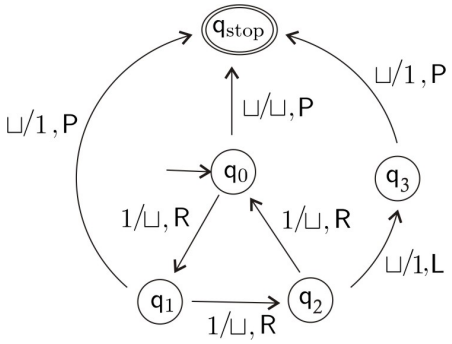
$$q_01^n \xrightarrow{*} \begin{cases} q_{da}\sqcup, & 3 \mid n \\ q_{ne}\sqcup, & 3 \nmid n. \end{cases}$$

Кажемо да конструисана машина *одлучује*, одн. *решава проблем припадања* језику $L = \{1^n \mid n \in \mathbb{N}, 3 \mid n\}$ над алфабетом $\{1\}$.

Пример 7. Посматрајмо израчунавања програма:

$$\begin{array}{llll} q_0\sqcup\sqcup \mathsf{P} q_{stop} & q_1\sqcup 1\mathsf{P} q_{stop} & q_2\sqcup 1\mathsf{L} q_3 & q_3\sqcup 1\mathsf{P} q_{stop} \\ q_01\sqcup \mathsf{R} q_1 & q_11\sqcup \mathsf{R} q_2 & q_21\sqcup \mathsf{R} q_0 & \end{array}$$

за улазе које чини једна реч алфабета $\{1\}$. Приметимо најпре да је дати програм некомплетан, јер недостаје петорка $q_31\cdots$ којом се налаже шта машина треба да ради ако у стању q_3 скенира ћелију у коју је уписано 1. Једноставно се може уочити да се оваква конфигурација никада не може достићи током извршавања датог програма за почетне конфигурације q_01^n, $n \geqslant 0$, па самим тим програм можемо по вољи комплетирати; на пример, додавањем петорке $q_311\mathsf{P} q_{stop}$. У наредним примерима, често ћемо изостављати петорке које немају ефекта на извршавање програма за изабрани улазни алфабет.



[41] $q_0\sqcup\sqcup \mathsf{L} q_2$ $q_01\sqcup \mathsf{R} q_1$
$q_1\sqcup 1\mathsf{L} q_1$ $q_11\sqcup \mathsf{R} q_0$

Написани програм, као и онај из претходног примера, рачуна остатак при дељењу броја n са 3 само што резултат приказује на други начин: штампа га на траци.[44]

$$\mathsf{q}_0 1^n \overset{*}{\to} \begin{cases} \mathsf{q}_{\text{stop}}\sqcup, & \text{остатак при дељењу } n \text{ са 3 је 0} \\ \mathsf{q}_{\text{stop}}1, & \text{остатак при дељењу } n \text{ са 3 је 1} \\ \mathsf{q}_{\text{stop}}11, & \text{остатак при дељењу } n \text{ са 3 је 2.} \end{cases}$$

Дакле, дата машина се зауставља за сваки улаз из Σ^* и у завршној конфигурацији глава је увек у истом стању – у стању q_{stop}.

Кажемо да дата машина *израчунава* функцију $f : \Sigma^* \to \Sigma^*$, дату са $f(1^n) = 1^{\mathrm{rm}(3,n)}$, где је $\mathrm{rm}(3,n)$ остатак при дељењу n са 3.

[44] Програм ће за улаз 1^n, $n \geqslant 0$, обрисати улаз и запамтити својим стањима остатак при дељењу броја n са 3, а затим ће на основу тога одштампати одговарајући резултат: ниједну јединицу ако до бланко симбола стигне у стању q_0, једну јединицу ако до бланко симбола стигне у стању q_1 и две јединице ако до бланко симбола стигне у стању q_2. Након штампањаући ће у стање q_{stop} и зауставити се.

Пример 8. Конструишимо Тјурингову машину која за две речи алфабета $\{a, b\}$ на прву од њих надовезују другу и зауставља се.

Прецизнија формулација задатка јесте: конструисати Тјурингову машину $\mathbb{T}$, чији је улазни алфабет $\{a, b\}$ и завршно стање q_{stop}, такву да за све $w_1, w_2 \in \{a, b\}^*$ важи: $\mathsf{q}_0 w_1 \sqcup w_2 \to^*_{\mathbb{T}} \mathsf{q}_{\text{stop}} w_1 w_2$.

Главни део конструкције јесте саставити програм, тј. дефинисати функцију транзиције (самим тим одредити Q и Γ). Притом, важно је имати на уму да $\mathbb{T}$ треба да споји две дате речи, па је потпуно неважно каква ће бити израчунавања уколико је на траци уписано више од две речи. Идеја која доводи до траженог програма је једноставна: реч w_2 треба транслирати улево за једно место.

Програм приказан на слици десно није комплетан, што не представља никакав суштински проблем и можемо га на произвољан начин комплетирати, јер то неће имати утицаја на израчунавање за било који улаз (w_1, w_2).

τ	q_0	q	q_a	q_b	q'	$\overleftarrow{\mathsf{q}}$
a	$\sqcup\mathsf{R}\mathsf{q}_0$	$\sqcup\mathsf{L}\mathsf{q}_a$				$a\mathsf{L}\overleftarrow{\mathsf{q}}$
b	$\sqcup\mathsf{R}\mathsf{q}_0$	$\sqcup\mathsf{L}\mathsf{q}_b$				$b\mathsf{L}\overleftarrow{\mathsf{q}}$
$\sqcup$	$\sqcup\mathsf{R}\mathsf{q}$	$\sqcup\mathsf{L}\mathsf{q}'$	$a\mathsf{R}\mathsf{q}_0$	$b\mathsf{R}\mathsf{q}_0$	$\sqcup\mathsf{L}\overleftarrow{\mathsf{q}}$	$\sqcup\mathsf{R}\mathsf{q}_{\text{stop}}$

Као илустрацију, наводимо израчунавање машине $\mathbb{T}$ за улаз (a, ba):

$$\begin{aligned} \mathsf{q}_0 a \sqcup ba &\to a\mathsf{q}_0 \sqcup ba \to a \sqcup \mathsf{q}ba \to a\mathsf{q}_b \sqcup \sqcup a \\ &\to ab\mathsf{q}_0 \sqcup a \to ab \sqcup \mathsf{q}a \to ab\mathsf{q}_a \sqcup \\ &\to aba\mathsf{q}_0 \sqcup \to aba \sqcup \mathsf{q}\sqcup \to aba\mathsf{q}'\sqcup \\ &\to ab\overleftarrow{\mathsf{q}}\, a \to a\overleftarrow{\mathsf{q}}\, ba \to \overleftarrow{\mathsf{q}}\, aba \to \overleftarrow{\mathsf{q}}\, \sqcup aba \to \mathsf{q}_{\text{stop}} aba \end{aligned}$$

Пример 9. Конструишимо Тјурингову машину такву да за сваку реч $w \in \{0, 1\}$ важи $q_0 w \to^*_{\mathbb{T}} q_{\text{stop}} w\#w$. Наводимо најпре уопштен опис траженог програма:

1. Нађи први бланко знак здесна и уместо њега упиши #.

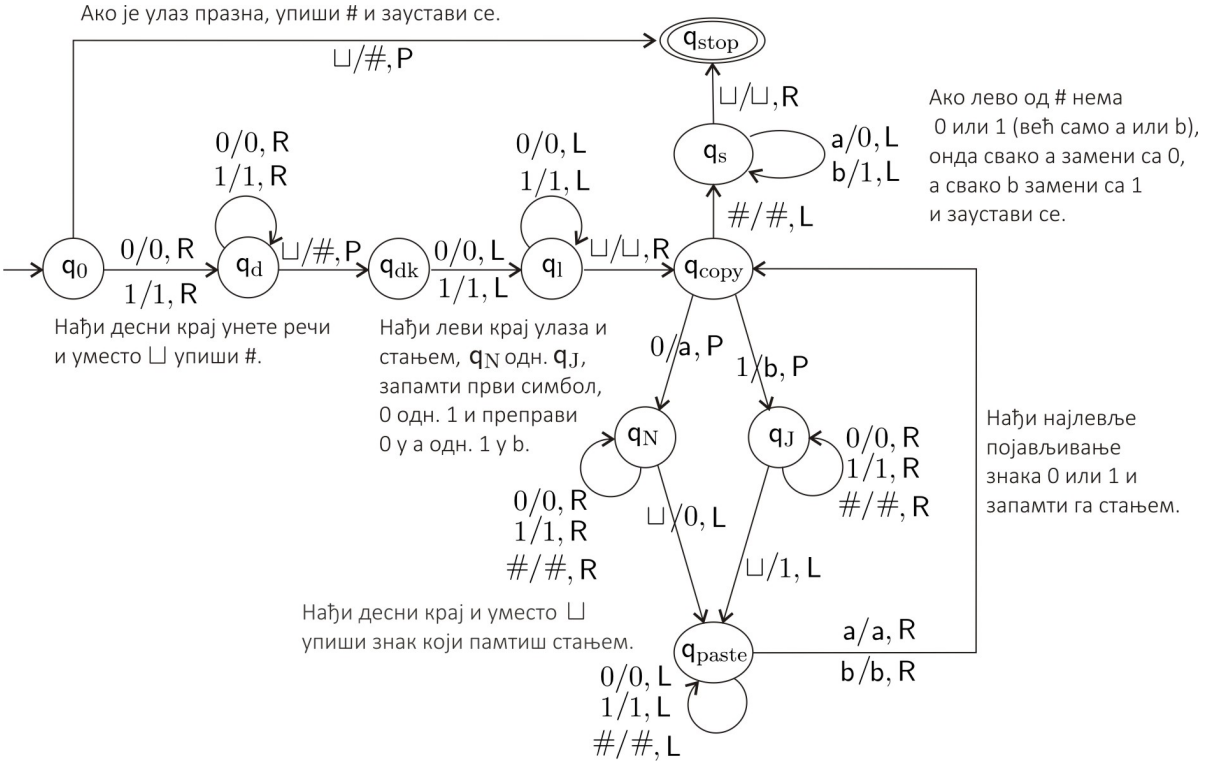
2. Понављај, док год је могуће следећи поступак:

нађи најлевљи симбол (лево од #) запамти га стањем и симбол 0 преправи у a, а 1 у b; затим симбол који памтиш упиши уместо првог бланко знака здесна (десно од #);

иначе пређи на наредни корак.

3. ако су са леве стране знака # уписани само симболи ⊔, a, b, онда симболе a преправи у 0, симболе b преправи у 1 и заврши у стању q_{stop} читајући први симбол слева различит од ⊔.

У алфабет траке смо, поред симбола # који налаже поставка задатка, укључили и помоћне симболе a и b.



Издвајамо неке карактеристичне кораке израчунавања конструисане машине за улаз 01:

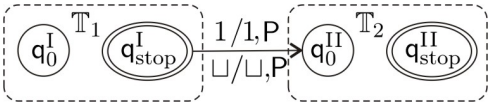
$$\begin{aligned}
&\mathsf{q}_0 01 \to \cdots \to 01\mathsf{q}_d\sqcup \to 01\mathsf{q}_{\text{dk}}\# \to \cdots \mathsf{q}_l 01\# \\
&\quad \to \mathsf{q}_{\text{copy}}01\# \to \mathsf{q}_N a1\# \to \cdots \to a1\#\mathsf{q}_N\sqcup \to a1\mathsf{q}_{\text{paste}}\#0 \to \cdots \\
&\quad \to a\mathsf{q}_{\text{copy}}1\#0 \to a\mathsf{q}_J b\#0 \to \cdots \to ab\#0\mathsf{q}_J\sqcup \to ab\#\mathsf{q}_{\text{paste}}01 \to \cdots \\
&\quad \to ab\mathsf{q}_{\text{copy}}\#01 \to a\mathsf{q}_s b\#01 \to \mathsf{q}_s a1\#01 \to \mathsf{q}_s \sqcup 01\#01 \\
&\quad \to \mathsf{q}_{\text{stop}}01\#01
\end{aligned}$$

ПРИМЕР 10. Конструишимо Тјурингову машину $\mathbb{T}$ такву да за $m, n \in \mathbb{N}$, $\mathsf{q}_0 1^m \sqcup 1^n \to^*_{\mathbb{T}} \mathsf{q}_{\text{stop}} 1^{m+_3 n}$. Тражену машину добијамо **надовезивањем**: $m +_3 n \stackrel{\text{def}}{=} \text{rm}(3, m+n)$

- машине $\mathbb{T}_{\text{I}}$ из Примера 8 која је прилагођена унарном алфабету $\{1\}$: $\mathsf{q}^{\text{I}}_0 1^m \sqcup 1^n \to^*_{\mathbb{T}_1} \mathsf{q}^{\text{I}}_{\text{stop}} 1^{m+n}$

- машине $\mathbb{T}_{\text{II}}$ из Примера 7: $\mathsf{q}_0^{\text{II}}1^n \to_{\mathbb{T}_2}^* \mathsf{q}_{\text{stop}}^{\text{II}}1^{\text{rm}(3,n)}$.

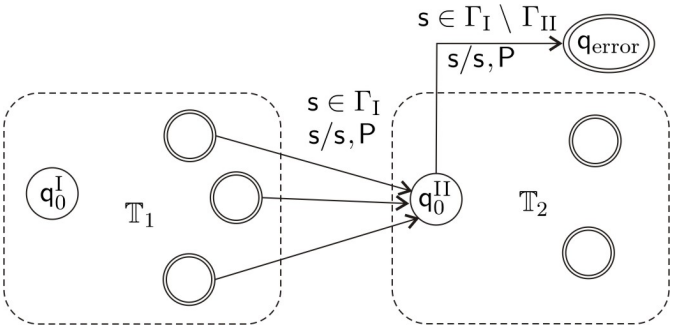
Без губљења општости претпостављамо да су сва стања машине $\mathbb{T}_{\text{I}}$ означена различито од стања машине $\mathbb{T}_{\text{II}}$. ТМ $\mathbb{T}$ ће прво извршавати наредбе машине $\mathbb{T}_{\text{I}}$, затим следеће две додатне наредбе, $\mathsf{q}_{\text{stop}}^{\text{I}}11\mathsf{P}\mathsf{q}_0^{\text{II}}$, $\mathsf{q}_{\text{stop}}^{\text{I}}\sqcup\sqcup\mathsf{P}\mathsf{q}_0^{\text{II}}$, и најзад наредбе машине $\mathbb{T}_{\text{II}}$.



Надовезивањем Тјурингових машина

$$\mathbb{T}_1 = (Q_{\text{I}}, \mathsf{q}_0^{\text{I}}, F_{\text{I}}, \Sigma_{\text{I}}, \sqcup, \Gamma_{\text{I}}, \tau_{\text{I}}) \text{ и } \mathbb{T}_2 = (Q_{\text{II}}, \mathsf{q}_0^{\text{II}}, F_{\text{II}}, \Sigma_{\text{II}}, \sqcup, \Gamma_{\text{II}}, \tau_{\text{II}}),$$

на очекивани начин, градимо нову машину $\mathbb{T} = \mathbb{T}_1\mathbb{T}_2$, чија су израчунавања добијена тако што се на израчунавање машине $\mathbb{T}_1$ надовезује израчунавање машине $\mathbb{T}_2$. Под претпоставком да је $Q_{\text{I}} \cap Q_{\text{II}} = \emptyset$ (што увек можемо постићи преименовањем стања), програм τ машине $\mathbb{T}$ треба да садржи све наредбе програма τ_{I} и τ_{II}, као и $\mathsf{q}_{\text{stop}}^{\text{I}}\mathsf{s}\mathsf{s}\mathsf{P}\mathsf{q}_0^{\text{II}}$, $\mathsf{s} \in \Gamma_{\text{I}}$. Да бисмо потпуно прецизирали поступак нодовезивања машина, по договору (само ако је потребно!) машини $\mathbb{T}$ додајемо још једно ново завршно стање $\mathsf{q}_{\text{error}}$ у које ће улазити када у стању q_0^{II} чита симбол који није у Γ_{II}. Овако добијени списак наредби није комплетан, али га можемо комплетирати на тривијалан начин.



Дакле,

$$\mathbb{T}_1\mathbb{T}_2 = (Q_{\text{I}} \cup Q_{\text{II}} \ (\cup\{\mathsf{q}_{\text{error}}\}), \mathsf{q}_0^{\text{I}}, F_{\text{II}} \ (\cup\{\mathsf{q}_{\text{error}}\}), \Sigma_{\text{I}}, \sqcup, \Gamma_{\text{I}} \cup \Gamma_{\text{II}}, \tau).$$

Надовезивање више од две Тјурингове машине, $\mathbb{T}_1, \mathbb{T}_2, \ldots \mathbb{T}_k$, дефинишемо индуктивно: $\mathbb{T}_1\mathbb{T}_2 \cdots \mathbb{T}_k = (\mathbb{T}_1\mathbb{T}_2 \cdots \mathbb{T}_{k-1})\mathbb{T}_k$. За машину $\mathbb{T}$ и $k \geqslant 1$, програм $\underbrace{\mathbb{T} \cdots \mathbb{T}}_{k \text{ пута}}$ означавамо $\mathbb{T}^k$.

Главни „ликови“ ове књиге јесу *функције* чије вредности може израчунавати нека Тјурингова машина, одн. *скупови* за које је *проблем припадања* одлучив неком Тјуринговом машином.

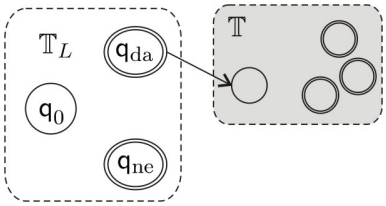
Дефиниција 2. *1) Језик $L \subseteq \Sigma^{*k}$ је* ТМ-***одлучив*** *ако постоји Тјурингова машина $\mathbb{T}$ са улазним алфабетом Σ таква да за све речи*

$w_1, \ldots, w_k \in \Sigma^*$:

$$\mathsf{q}_0 w_1 \sqcup \cdots \sqcup w_k \to_{\mathbb{T}}^* \begin{cases} \mathsf{q}_{\mathrm{da}} \sqcup, & \overline{w} \in L \\ \mathsf{q}_{\mathrm{ne}} \sqcup, & \overline{w} \notin L. \end{cases}$$

2) Функција $f : \Sigma^{*k} \to \Gamma^*$ је ТМ-***израчунљива*** *ако постоји Тјурингова машина $\mathbb{T}$ са улазним алфабетом Σ таква да за све речи $w_1, \ldots, w_k \in \Sigma^*$: $\mathsf{q}_0 w_1 \sqcup \cdots \sqcup w_k \to_{\mathbb{T}}^* \mathsf{q}_{\mathrm{stop}} f(\overline{w})$.*

У наредним поглављима, често ћемо конструисати ТМ подразумевајући да оне као улазе добијају само елементе неког ТМ-одлучивог скупа $L \subseteq \Sigma^{*k}$, тј. водећи рачуна само о улазима из L, и занемарујући улазе који не припадају L. У таквим случајевима ми заправо конструишемо део $\mathbb{T}$ машине приказане на слици десно, подразумевајући да је тај део надовезан на стање q_{da} машине $\mathbb{T}_L$ која одлучује L.



Пример 11. Према ранијем договору, улазе сачињене од $k > 1$ речи улазног алфабета Σ уносимо на траку тако што речи раздвајамо једним бланко знаком. Уместо тога, некад је згодно улазни алфабет проширити знаком који не припада Σ, на пример знаком #, и k-торке речи из Σ^* идентификовати са речима из $\Sigma \cup \{\#\}$ које садрже тачно $k-1$ знакова #. То нам омогућава чињеница да је језик $\{w \in (\Sigma \cup \{\#\})^* \mid |w|_\# = k-1\}$ ТМ-одлучив.[48] Тако, на пример, да бисмо показали да неки језик $L \subseteq \Sigma^{*2}$ јесте ТМ-**одлучив** довољно је конструисати Тјурингову машину $\mathbb{T}$ такву да за све $w_1, w_2 \in \Sigma^*$:

$$\mathsf{q}_0 w_1 \# w_2 \to_{\mathbb{T}}^* \begin{cases} \mathsf{q}_{\mathrm{da}} \sqcup, & (w_1, w_2) \in L \\ \mathsf{q}_{\mathrm{ne}} \sqcup, & (w_1, w_2) \notin L. \end{cases}$$

Јасно, $L \subseteq \Sigma^* \times \Sigma^*$ је ТМ-одлучив акко је језик

$$\{w_1 \# w_2 \mid (w_1, w_2) \in L\}$$

ТМ-одлучив.

Слично, функција $f : \Sigma^{*2} \to \Gamma^*$ је ТМ-израчунљива ако постоји ТМ $\mathbb{T}$ таква да за све $w_1, w_2 \in \Sigma^*$: $\mathsf{q}_0 w_1 \# w_2 \to_{\mathbb{T}}^* \mathsf{q}_{\mathrm{stop}} f(w_1, w_2)$.

Пример 12. Бинарне записе природних бројева не чине све речи из $\{0,1\}^*$, већ само речи из

$$B = \{0\} \cup \{1w \mid w \in \{0,1\}^*\} \subseteq \{0,1\}^*.$$

Будући да је B ТМ-одлучив, ако треба конструисати Тјурингову машину која израчунава вредности неке функције чији су аргументи $(w_1, \ldots, w_k)$ бинарни записи природних бројева, узимамо у обзир само улазе такве да $w_i \in B$, $1 \leqslant i \leqslant k$.

[48] $|w|_{\mathsf{s}}$ означава број појављивања симбола s у речи w.

Задаци

1. Доказати да су следећи језици (над одговарајућим алфабетима) TM-одлучиви:
а) $L = \{w \in \{0,1\}^* \mid w \text{ је палиндром}\}$;
б) $L = \{w\#w \mid w \in \{0,1\}^*\} \subseteq \{0,1,\#\}^*$;
в) $L = \{a^k b^\ell c^m \mid k,\ell,m \in \mathbb{N}, m = k \cdot \ell\} \subseteq \{a,b,c\}^*$;[49]
г) $L = \{1^{n^2} \mid n \in \mathbb{N}\} \subseteq \{1\}^*$.

[49] На пример, $a^2b^3c^6 \in L$; $a^2b^3c^5 \notin L$.

2. Доказати да је TM-израчунљива функција $f : \{1\}^* \times \{1\}^* \to \{1\}^*$ дата са:
а) $f(1^m, 1^n) = 1^{m+_2 n}$, $m,n \geq 0$;
б) $f(1^m, 1^n) = \begin{cases} 1^{m-n}, & \text{ако је } m \geqslant n, \\ \varepsilon, & \text{ако је } m < n, \end{cases}$
б) $f(1^m, 1^n) = 1^{m \cdot n}$, $m,n \geq 0$;
в) $f(1^m, 1^n) = 1^{\mathrm{NZD}(m,n)}$, $m,n \geq 0$.

3. а) Доказати да је TM-одлучив језик $Z = \{1^n \mid n \geqslant 1\} \cup \{-1^n \mid n > 0\}$ алфабета $\{-,1\}$.
б) Ако природан број $n \in \mathbb{N}$ идетификујемо са речју 1^n, а негативан број $-n$, $n \in \mathbb{N}^+$, са речју -1^n, конструисати Тјурингову машину $\mathbb{T}$ која сабира целе бројеве.[50]

[50] На пример:
$\mathsf{q}_0 11 \sqcup -111 \to^*_{\mathbb{T}} \mathsf{q}_{\mathsf{stop}} - 1$,
$\mathsf{q}_0 - 11 \sqcup 111 \to^*_{\mathbb{T}} \mathsf{q}_{\mathsf{stop}} 1$,
$\mathsf{q}_0 11 \sqcup -11 \to^*_{\mathbb{T}} \mathsf{q}_{\mathsf{stop}} \sqcup$, итд.

4. Доказати да је функција $f : \{1\}^* \to \{0,1\}^*$ дата са $f(1^n) = \mathrm{bin}(n)$, $n \geq 0$, где је $\mathrm{bin}(n)$ бинарни запис броја n, TM-израчунљива.

5. Конструисати TM која сабира природне бројеве задате у бинарном запису.

6. а) Доказати да је TM-одлучив језик $L = \{1^k x^n \mid k \geqslant 0, n > 0\}$ алфабета $\{1,x\}$.
б) Конструисати Тјурингову машину $\mathbb{T}$ такву да за све $k \geqslant 0$, $n > 0$ важи $\mathsf{q}_0 1^k x^n \to^*_{\mathbb{T}} \mathsf{q}_{\mathsf{stop}} 1^{k \cdot n} x^{n-1}$.

7. Доказати да је TM-израчунљива функција $f : \{a,b\}^{*2} \to \Sigma^*$ дата са $f(w_1,w_2) = w_2^{|w_1|}$. [51]

[51] Подсећамо да је $|w_1|$ дужина речи w_1, па је $w_2^{|w_1|} = \underbrace{w_2 \cdots w_2}_{|w_1| \text{ пута}}$.

8. (а) Доказати да је језик $\{w \in \{0,1,\#\}^* \mid |w|_\# = 2\}$ TM-одлучив.
(б) Конструисати Тјурингову машину која израчунава функцију $f : \{0,1,\#\}^* \to \{\mathsf{Y},\mathsf{E},\mathsf{S},\mathsf{N},\mathsf{O}\}^*$,

$$f(w) = \begin{cases} \mathsf{q}_{\mathsf{stop}}\mathsf{YES} & |w|_\# = 2, \\ \mathsf{q}_{\mathsf{stop}}\mathsf{NO}, & |w|_\# \neq 2. \end{cases}$$

9. Конструисати Тјурингову машину $\mathbb{LEX}$ која израчунава функцију $f : \{0,1\}^* \to \{0,1\}^*$, где је $f(w)$ једнако $<_{\mathrm{lex}}$-непосредном следбенику речи w. Дакле, за сваку реч $w \in \{0,1\}$ треба да важи $\mathsf{q}_0 w \to^*_{\mathbb{LEX}} \mathsf{q}_{\mathsf{stop}} w'$, где је w' лексикографски следбеник речи w.

На пример, ако је $w = 001011011$, онда је $w' = (001011011)' = 001011100$.

Регистар машине

Регистар машине су апстрактне машине „новије генерације“. Замислимо траку која је неограничена само са једне стране и издељена на *регистре* означене $\mathsf{R}_1, \mathsf{R}_2, \mathsf{R}_3, \ldots$ У сваки регистар може бити уписана читава реч унапред изабраног алфабета $\Sigma = \{\mathsf{s}_1, \ldots, \mathsf{s}_m\}$. Уз траку дат је и тзв. *бројач*, у који се уноси природан број различит од нуле – редни број инструкције одговарајућег програма. Поред тога, замислимо и да постоји механизам који може да чита садржај регистра и да извршава једну од инструкција RM-**програма**, тј. коначног низа инструкција нумерисаних бројевима од 1 до неког n: $I_1, \ldots, I_n$, при чему је свака инструкција I_i следећег облика:

$i.\quad R_k^{+\mathsf{s}} \mid \ell \qquad (\mathsf{s} \in \Sigma, \ell \geqslant 1)$

или

$$i.\quad R_k^{-} \left| \begin{array}{ll} \varepsilon & \ell_0 \\ \mathsf{s}_1 & \ell_1 \\ \vdots & \\ \mathsf{s}_m & \ell_m \end{array} \right. \qquad (\ell_0, \ell_1 \ldots, \ell_m \geqslant 1)$$

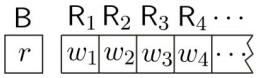
Значење: Допиши симбол s на десни крај речи која је уписана у регистар R_k и пређи на извршавање инструкције I_ℓ; број ℓ називамо *параметром преласка*.

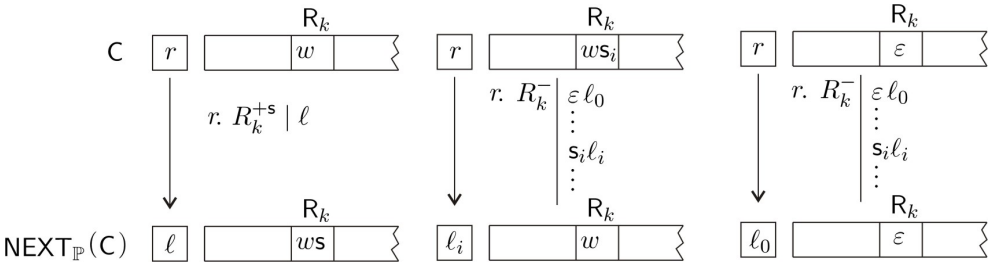
Значење: Ако је у R_k уписана празна реч, пређи на извршавање инструкције I_{ℓ_0}; а иначе, обриши последњи симбол речи из R_k и ако је обрисани симбол s_j пређи на извршавање инструкције I_{ℓ_j}; бројеве $\ell_0, \ell_1, \ldots, \ell_m$ називамо *параметрима преласка* ове инструкције.

Под **конфигурацијом** над Σ подразумевамо низ:

$$(\mathsf{C}) \quad r;\, w_1,\, w_2,\, w_3,\, \ldots$$

чији је први члан природни број различит од нуле (уписан у бројач), а остали чланови су речи из Σ^* (уписане редом у регистре $\mathsf{R}_1, \mathsf{R}_2, \ldots$). Конфигурација (C) је **завршна конфигурација** програма $\mathbb{P}$ ако $\mathbb{P}$ не садржи инструкцију чији је редни број r, тј. ако је $r > n$. Ако (C) није завршна конфигурација програма $\mathbb{P}$, онда је овим програмом одређен јединствени следбеник $\mathsf{NEXT}_{\mathbb{P}}(\mathsf{C})$, тј. конфигурација која се добија извршавањем инструкције I_r у складу са њеним значењем:

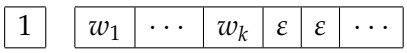




Рад RM-програма $\mathbb{P}$: $I_1, I_2, \ldots, I_n$ посматраћемо искључиво за **почетне конфигурације** у којима је у бројачу уписан број 1 (извршавање програма почиње инструкцијом која је прва наведена у $\mathbb{P}$) и у коначно много регистара су уписане неке речи алфабета Σ, а у свим осталим је аутоматски уписана празна реч. Дакле, почетне

конфигурације су облика:

$$(\mathsf{C}_0) \quad 1;\, w_1,\, \ldots,\, w_k,\, \varepsilon,\, \varepsilon,\, \varepsilon,\, \cdots$$



при чему се k назива *дубина улаза*. Сваки RM-програм $\mathbb{P}$ за улаз $(w_1, \ldots, w_k)$ генерише један низ конфигурација:

$$\mathsf{C}_0 \to \mathsf{NEXT}_{\mathbb{P}}(\mathsf{C}_0) \to \mathsf{NEXT}_{\mathbb{P}}(\mathsf{NEXT}_{\mathbb{P}}(\mathsf{C}_0)) \to \cdots$$

који је:

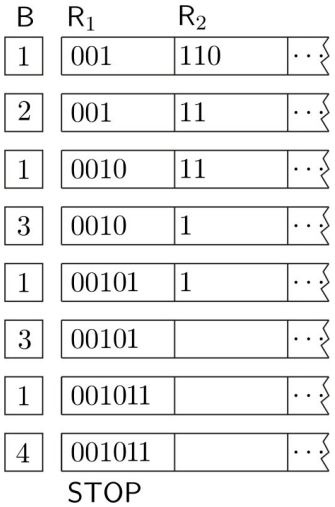
- *коначан* ако се у низу појави завршна конфигурација за програм $\mathbb{P}$; тада пишемо $\mathbb{P}(w_1, \ldots, w_k) \downarrow$ и кажемо *програм* $\mathbb{P}$ *се зауставља (конвергира) за улаз* $(w_1, \ldots, w_k)$; или
- *бесконачан* ако се у низу никада не појављује завршна конфигурација; тада пишемо $\mathbb{P}(w_1, \ldots, w_k) \uparrow$ и кажемо *програм* $\mathbb{P}$ *се не зауставља (дивергира) за улаз* $(w_1, \ldots, w_k)$.

Ако $\mathbb{P}(w_1, \ldots, w_k) \downarrow$, коначан низ конфигурација $\mathsf{C}_0, \mathsf{C}_1, \ldots, \mathsf{C}_d$, где је:

- C_0 почетна конфигурација одређена улазом $(w_1, \ldots, w_k)$,
- $\mathsf{C}_{i+1} = \mathsf{NEXT}_{\mathbb{P}}(\mathsf{C}_i)$, $1 \leqslant i < d$; пар суседних конфигурација називамо *рачунским кораком*; и
- C_d завршна конфигурација за $\mathbb{P}$,

називамо **израчунавањем** програма $\mathbb{P}$ за улаз $(w_1, \ldots, w_k)$ и пишемо $\mathsf{C}_0 \to^*_{\mathbb{P}} \mathsf{C}_d$.

Будући да је сваки RM-програм $\mathbb{P}$ коначан низ инструкција наведеног облика, у њему се помиње само коначно много регистара и садржаји само тих регистара могу бити измењени током извршавања програма $\mathbb{P}$. Најмањи природан број $\|\mathbb{P}\|$ такав да се у програму $\mathbb{P}$ помињу само регистри R_i, $i \leqslant \|\mathbb{P}\|$, назива се *дубина програма* $\mathbb{P}$. Узимајући у обзир да је на почетку у свим регистрима почев од неког уписана празна реч, закључујемо да ће и током читавог израчунавања садржаји само коначно много регистара бити различити од празне речи. Зато, све достижне конфигурације током рада програма $\mathbb{P}$ можемо означавати коначним низом $i, w_1, \ldots, w_d$, $d = \max\{n, \|\mathbb{P}\|\}$, где је n *дубина улаза*, а $\|\mathbb{P}\|$ *дубина програма*.



Пример 13. Нека је $\mathbb{P}$ програм над алфабетом $\{0, 1\}$:

1.	R_2^-	ε	4
		0	2
		1	3

2. $\mathsf{R}_1^{+0} \mid 1$ 3. $\mathsf{R}_1^{+1} \mid 1$

Овај програм за улаз $(001, 110)$ генерише следећи низ конфигурација:

$$1;001,110 \to 2;001,11 \to 1;0010,11 \to 3;0010,1 \to 1;00101,1$$
$$\to 3;00101,\varepsilon \to 1;001011,\varepsilon \to 4;001011,\varepsilon$$

Како је последња конфигурација наведеног низа уједно и завршна конфигурација за програм $\mathbb{P}$, следи да $\mathbb{P}(001,110)\downarrow$.

Није тешко уочити да ће се израчунавање програма $\mathbb{P}$ зауставити за сваки улаз $(w_1,w_2,\ldots,w_k)$, $k\geqslant 2$: $\mathbb{P}(w_1,w_2,\ldots,w_k)\downarrow$ и у завршној конфигурацији, садржај регистра R_1 ће бити реч $w_1w_2^-$, где је w_2^- реч добијена од w_2 записивањем њених симбола у супротном поретку, садржај регистра R_2 ће бити празна реч, а свих осталих као и у почетној конфигурацији. За свако $k>2$ и све $(w_1,w_2,\ldots,w_k)$,

$$1;w_1,w_2,\ldots,w_k\to^*_{\mathbb{P}} 4;w_1w_2^-,\varepsilon,w_3,\ldots,w_k.$$

Специјално, за све $w\in\Sigma^*$, $\mathbb{P}(w)\downarrow$ и $1;w\to^*_{\mathbb{P}} 4;w$.

Пример 14. Следећи програм $\mathbb{P}$, над $\{a,b\}$ садржи само једну инструкцију :

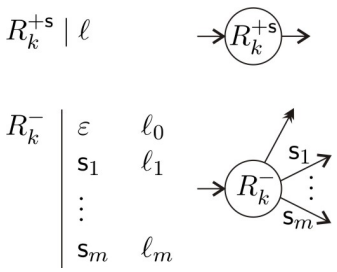
1.	R_1^-	ε	1
		a	1
		b	2

$\mathbb{P}$ се не зауставља за сваки улаз. Ако је на почетку у регистар R_1 уписана реч која не садржи b, онда се овај програм не зауставља; на пример $\mathbb{P}(aaa)\uparrow$: $1;aaa\to 1;aa\to 1;a\to 1;\varepsilon\to 1;\varepsilon\to\cdots$ Ако у R_1 на почетку унесемо реч која садржи b, тј. реч облика wba^n, за неко $w\in\{a,b\}^*$ и неко $n\geqslant 0$, онда $1;wba^n\to^*_{\mathbb{P}} 2;w$.

Пример 15. Није тешко саставити бројне програме, над било којим алфабетом Σ, који се неће заустављати ни за какве улазе. Наводимо два таква програма за $\Sigma=\{a,b\}$:

$\mathbb{P}_1$					$\mathbb{P}_2$			
1.	R_1^-	ε	1		1.	R_1^{+a}	1	
		a	1					
		b	1					

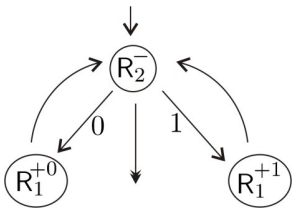
Погодно је RM-програме приказивати посебном врстом (коначних оријентисаних) графова састављених од делова, приказаних на слици десно, који одговарају инструкцијама RM-програма. Редослед којим инструкције треба извршавати одређујемо: (1) избором једне *улазне стрелице*, која долази из „спољашњости“ и показује само на један чвор, и (2) усмеравањем стрелица које излазе из чворова, при чему свака стрелица може бити упућена на било који чвор графа укључујући и чвор из кога полази, а може остати и „слободна“, тј. неповезана ни са једним чвором – овакву стрелицу називамо *излазном стрелицом*. Излазне стрелице заправо указују на заустављање извршавања програма.



Пример 16. Графички приказ RM-програма $\mathbb{P}$ из примера 13:

1.	R_2^-	ε	4	2.	R_1^{+0}	1	3.	R_1^{+1}	1
		0	2						
		1	3						

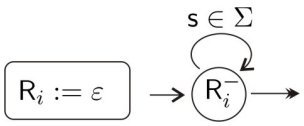
дат је на слици десно.



Пример 17. Наводимо неколико RM-програма, над произвољним алфабетом $\Sigma = \{\mathsf{s}_1, \ldots, \mathsf{s}_m\}$, које ћемо у наставку користити приликом састављања неких сложенијих RM-програма.

Програм $\mathsf{R}_i := \varepsilon$ брише садржај регистра R_i.

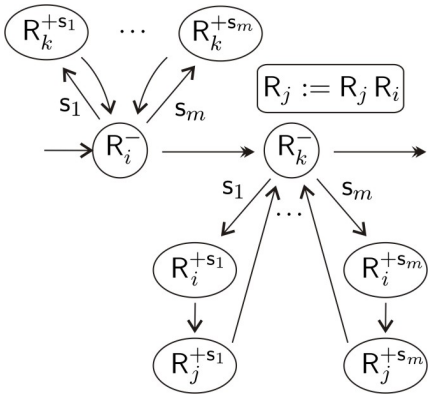
1.	R_i^-	ε	2
		s_1	1
		$\vdots$	
		s_m	1



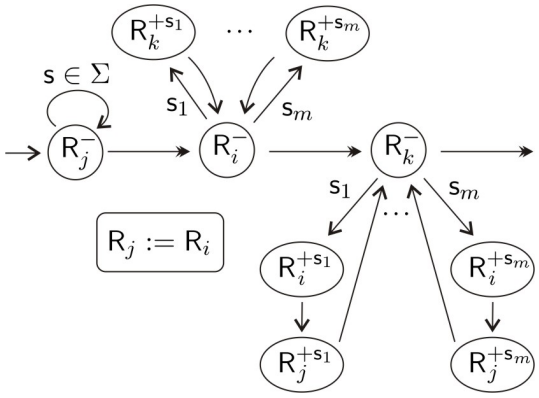
Програм $\mathsf{R}_j := \mathsf{R}_j\mathsf{R}_i$ (приказан десно) на садржај регистра R_j дописује (здесна) садржај регистра R_i, уз помоћ регистра R_k у коме је празна реч и на почетку и на крају. Након извршавања овог програма, садржај регистра R_i остаје исти као што је био на почетку. Експлицитно наводимо наредбе овог програма за алфабет $\{a, b\}$:

1. R_i^-	ε 4	2. R_k^{+a} | 1	3. R_k^{+a} | 1	4. R_k^-	ε 9
	a 2				a 5
	b 3				a 7

5. R_i^{+a} | 6 6. R_j^{+a} | 4 7. R_i^{+b} | 8 8. R_j^{+b} | 4



Програм $\mathsf{R}_j := \mathsf{R}_i$ (приказан десно) садржај регистра R_i копира у R_j уз помоћ регистра R_k у коме је празна реч и на почетку копирања и након копирања; после копирања, садржај регистра R_i остаје исти као што је био на почетку. Није тешко уочити да је програм $\mathsf{R}_j := \mathsf{R}_i$ заправо добијен *надовезивањем* претходна два програма, $\mathsf{R}_j := \varepsilon$ и $\mathsf{R}_j := \mathsf{R}_j\mathsf{R}_i$. Наравно, надовезивање два (или више) програма не значи просто спајање тих програма, већ подразумева одговарајуће измене инструкција. Прецизније, неопходно је у инструкцијама изменити параметре прелаза. Општи поступак надовезивања програма описаћемо у наставку.



Да бисмо поједноставили теоријска разматрања о RM-програмима, углавном ћемо посматрати оне програме који су у тзв. *стандардној форми*. Програм $\mathbb{P} = I_1, \ldots, I_n$ је у стандардној форми уколико за сваки параметар прелаза ℓ који се појављује у некој инструкцији I_i, $1 \leqslant i \leqslant n$, важи $1 \leqslant \ell \leqslant n+1$. Другим речима, извршавање програма $\mathbb{P}$ се зауставља само када је у бројачу уписан број $n+1$.

Два RM-програма су **еквивалентна** ако за све улазе $(w_1, \ldots, w_k)$, $k \geqslant 1$, генеришу иста израчунавања. Сваки RM-програм се једноставно може прерадити у еквивалентан RM-програм који је у стандардној форми. Уместо строгог доказа, наводимо само **поступак стандардизације** произвољног програма $\mathbb{P} = I_1, \ldots, I_n$. У свакој инструкцији I_i која садржи параметар прелаза ℓ различит од $1, \ldots, n$, тај параметар ℓ замењујемо бројем $n+1$, а ако инструкција I_i не садржи такав параметар, онда I_i остаје не измењена.

Нови програм $\mathbb{P}'$ добијен из $\mathbb{P}$ након оваквих измена, јесте у стандардној форми и очигледно је еквивалентан полазном програму.

Ако су $\mathbb{P}_1$ и $\mathbb{P}_2$ два RM-програма у стандардној форми:

$$\mathbb{P}_1 : 1.\ I_1\ \cdots\ n_1.\ I_{n_1}; \quad \mathbb{P}_2 : 1.\ J_1\ \cdots\ n_2.\ J_{n_2},$$

онда **надовезивањем** $\mathbb{P}_2$ на $\mathbb{P}_1$ добијамо програм

$$\mathbb{P}_1\mathbb{P}_2 : 1.\ I_1\ \cdots\ n_1.\ I_{n_1} \quad n_1+1.\ J'_1\ \cdots\ n_2+n_1.\ J'_{n_2},$$

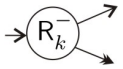
при чему је свака инструкција J'_i добијена из J_i тако што се параметри прелаза увећавају за n_1. Програме $\mathbb{P}_1$ и $\mathbb{P}_2$ називамо *потпрограмима* програма $\mathbb{P}_1\mathbb{P}_2$. Аналогно дефинишемо надовезивање три и више програма. На пример, $\mathbb{P}_1\mathbb{P}_2\mathbb{P}_3$ је програм добијен надовезивањем програма $\mathbb{P}_3$ на $\mathbb{P}_1\mathbb{P}_2$.

RM-програми за унарни улазни алфабет. У наставку одељка, посебну пажњу посвећујемо RM-програмима за унарни алфабет $\Sigma = \{1\}$. Подсећамо да скуп речи унарног алфабета $\Sigma = \{1\}$ на природан начин идентификујемо са скупом природних бројева $\mathbb{N}$, тј. реч 1^n идентификујемо са n, па ћемо зато уместо 1^n у наставку писати n. Инструкције RM-програма се за алфабет $\{1\}$ прилично поједностављују.

- Инструкција $\mathsf{R}_k^+ \mid \ell$ значи да се садржају регистра R_k додаје 1 (тј. одређује се следбеник текућег садржаја) и прелази се на извршавање инструкције чији је редни број ℓ.

- Инструкцију

$$\begin{array}{c|cc} \mathsf{R}_k^- & \varepsilon & \ell_0 \\ & 1 & \ell_1 \end{array}$$



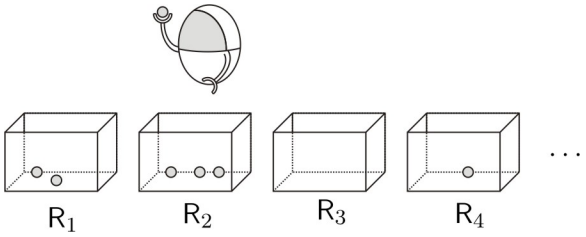
у наставку краће записујемо $\mathsf{R}_k^- \mid \ell_0, \ell_1$. Ова инструкција значи: ако је садржај регистра R_k једнак[61] 0 прелази се на извршавање инструкције чији је редни број ℓ_0, а ако је садржај регистра R_k различит од 0 умањује се за 1 (одређује се претходник) и прелази се на извршавање инструкције чији је редни број ℓ_1.

[61] Подсећамо да 0 означава 1^0, тј. празну реч.

За алфабет $\{1\}$, конфигурације су заправо низови природних бројева чији су сви чланови почев од неког једнаки нули. Ако Conf означава скуп свих конфигурација траке за алфабет $\{1\}$, сваки RM-програм $\mathbb{P}$ одређује функцију $\mathsf{NEXT}_{\mathbb{P}} : \mathrm{Conf} \to \mathrm{Conf}$:

$$\mathsf{NEXT}_{\mathbb{P}}(i; r_1, \ldots, r_k, \ldots, r_d, 0\cdots) = \begin{cases} (\ell; r_1, \ldots, r_k+1, \ldots, r_d, 0\cdots), & i\text{-та инструкција програма } \mathbb{P} \text{ је } R_k^+ \mid \ell, \\ (\ell_0; r_1, \ldots, r_k, \ldots, r_d, 0\cdots), & i\text{-та инструкција програма } \mathbb{P} \text{ је } R_k^- \mid \ell_0, \ell_1 \text{ и } r_k = 0, \\ (\ell_1; r_1, \ldots, r_k-1, \ldots, r_d, 0\cdots), & i\text{-та инструкција програма } \mathbb{P} \text{ је } R_k^- \mid \ell_0, \ell_1 \text{ и } r_k \neq 0, \\ (i; r_1, \ldots, r_k, \ldots, r_d, 0\cdots), & i\text{-та инструкција програма } \mathbb{P} \text{ не постоји.} \end{cases}$$

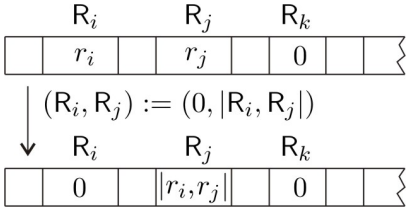
НАПОМЕНА. Ако регистре замислимо као кутије, при чему у сваку кутију може да стане произвољно (али коначно) много куглица, састављање RM-програма над унарним улазним алфабетом можемо схватити као писање програма за машину која може да убаци једну куглицу у било кутију и пређе на одговарајућу инструкцију (тј. може да изврши $\mathsf{R}_k^{+1} \mid \ell$) или да утврди да ли је кутија непразна и ако јесте из ње да извади једну куглицу и пређе на одговарајућу инструкцију, а ако је празна пређе на неку другу (тј. може да изврши $\mathsf{R}_k^{-} \mid \ell_0, \ell_1$).

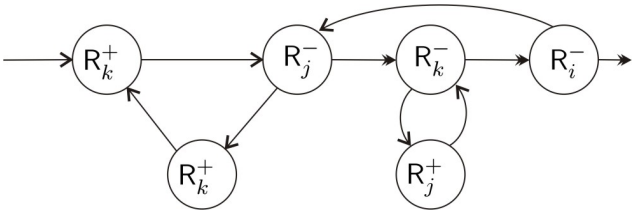


ПРИМЕР 18. Бијекцију $|\cdot,\cdot| : \mathbb{N} \times \mathbb{N} \overset{1-1}{\underset{\text{на}}{\rightarrow}} \mathbb{N}^+$, $|x_1, x_2| = 2^{x_1}(2x_2 + 1)$ користили смо за кодирање коначних низова природних бројева (стр. 10). Нека су $\mathrm{hd}, \mathrm{tl} : \mathbb{N} \to \mathbb{N}$ функције које одређују инверзну функцију ове бијекције: за $n \in \mathbb{N}^+$, $|\mathrm{hd}(n), \mathrm{tl}(n)| = n$, уз договор да је $\mathrm{hd}(0) = \mathrm{tl}(0) = 0$.

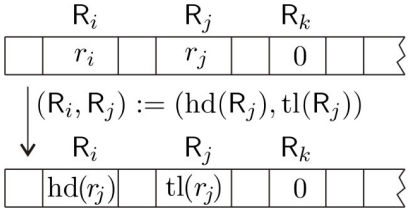
$\mathrm{hd}(n) =$ први члан низа чији је кôд n
$\mathrm{tl}(n) =$ кôд репа низа чији је кôд n

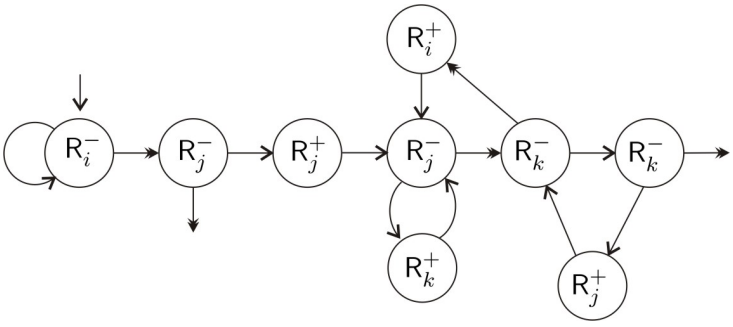
Саставићемо два RM-програма који се односе на ове функције. Први од њих је програм $(\mathsf{R}_i, \mathsf{R}_j) := (0, |\mathsf{R}_i, \mathsf{R}_j|)$ који израчунава $|r_i, r_j|$, где су r_i и r_j почетни садржаји редом регистара R_i и R_j, при чему се користи помоћни регистар R_k чији је садржај иницијално једнак 0, и такав ће бити и на крају израчунавања када ће садржаји регистара R_i и R_j редом бити једнаки 0 и $|r_i, r_j|$. Овај програм је приказан на наредној слици.





Други програм означавамо $(\mathsf{R}_i, \mathsf{R}_j) := (\mathrm{hd}(\mathsf{R}_j), \mathrm{tl}(\mathsf{R}_j))$. Претпостављајућу да је јасно шта овај програм треба да ради, наводимо само његов графички приказ.

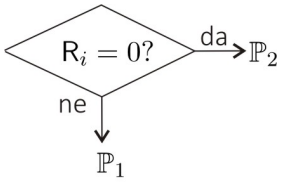




ПРИМЕР 19. Показаћемо да приликом састављања RM-програма можемо користити и тзв. наредбе *условног преласка* облика

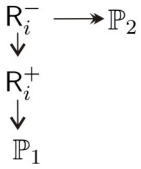
$$(*) \qquad \text{if } R_i = 0? \text{ then } \mathbb{P}_1 \text{ else } \mathbb{P}_2,$$

где су $\mathbb{P}_1$ и $\mathbb{P}_2$ неки RM-програми. Програмом $(*)$ се најпре испитује да ли је садржај r_i регистра R_i једнак нули или није; у потврдном случају, $r_i = 0$, извршава се програм $\mathbb{P}_2$ и програм $(*)$ се зауставља ако се $\mathbb{P}_2$ заустави, а у одречном случају, $r_i \neq 0$, извршава се програм $\mathbb{P}_1$ и програм $(*)$ се зауставља ако се $\mathbb{P}_1$ заустави.



Ако је $\mathbb{P}_1 = I_1, \ldots, I_{n+1}$ и $\mathbb{P}_2 = J_1, \ldots, J_{n+1}$, онда је $(*)$ следећи програм:

1. $R_i^- \mid n_1 + 3, 2$
2. $R_i^+ \mid 3$
3. I'_1
$\vdots$
$n_1 + 2$. I'_{n_1}
$n_1 + 3$. J'_1
$\vdots$
$n_1 + n_2 + 3$. J'_{n_2}

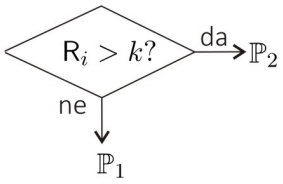


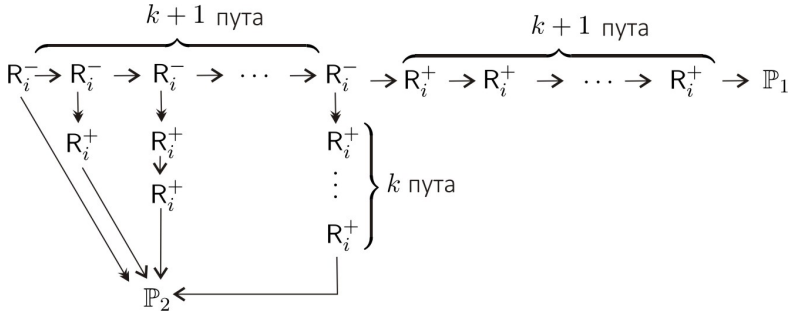
I'_i је наредба добијена из I_i тако што је сваки параметар прелаза ℓ промењен на следећи начин: ако је $1 \leqslant \ell \leqslant n_1$, онда је ℓ замењено са $\ell + 2$, ако је $\ell > n_1$, онда је ℓ замењено са $n_1 + n_2 + 4$. Аналогно, J'_i је наредба добијена из J_i тако што је сваки параметар прелаза ℓ промењен на следећи начин: ако је $1 \leqslant \ell \leqslant n_2$, онда је ℓ замењено са $\ell + n_1 + 2$, ако је $\ell > n_2$, онда је ℓ замењено са $n_1 + n_2 + 4$.

На слици испод дат је приказ сличног програма

$$\text{if } R_i > k? \text{ then } \mathbb{P}_1 \text{ else } \mathbb{P}_2,$$

где је k неки фиксирани природни број.

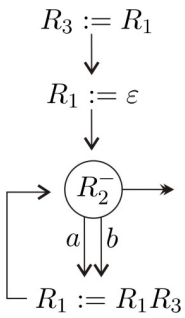




RM-програми су посебно погодни за израчунавање функција са више аргумената. Ако се извршавање програма $\mathbb{P}$ за улаз $(w_1, \ldots, w_k)$ зауставља, тј. $\mathbb{P}(w_1, \ldots, w_k)\downarrow$, по договору, под **резултатом** израчунавања подразумевамо реч v која је садржај првог регистра R_1 у завршној конфигурацији; пишемо $\mathbb{P}(w_1, \ldots, w_k) \downarrow v$.

Дефиниција 3. *Функција $f : \Sigma^{*k} \to \Sigma^*$ је* RM-***израчунљива***, *ако постоји* RM-*програм $\mathbb{P}$ такав да за све $w_1, \ldots, w_k \in \Sigma^*$ важи $\mathbb{P}(w_1, \ldots, w_k) \downarrow f(w_1, \ldots, w_k)$.*

Пример 20. Програм приказан на слици десно потврђује да је, за $\Sigma = \{a, b\}$, функција $f : \Sigma^* \times \Sigma^* \to \Sigma^*$, дата са $f(w_1, w_2) = w_1^{|w_2|}$ једна RM-израчунљива функција. Истичемо да су RM-програми из примера 17 коришћени као потпрограми.



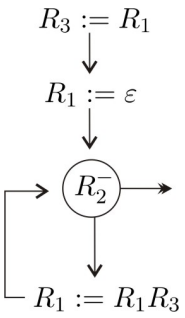
Идентификујући $\mathbb{N}$ са $\{1\}^*$, и природан број n са 1^n, према претходној дефиницији, функција $f : \mathbb{N}^k \to \mathbb{N}$ је RM-израчунљива,

ако постоји RM-програм $\mathbb{P}$ такав да за све $n_1, \ldots, n_k \in \mathbb{N}$ важи $\mathbb{P}(n_1, \ldots, n_k) \downarrow f(n_1, \ldots, n_k)$. У наставку ћемо RM-програме користити пре свега за израчунавање вредности функција над природним бројевима, па зато само у овом контексту прецизирамо појам RM-одлучивости.

Дефиниција 4. *Скуп $A \subseteq \mathbb{N}^k$ је* RM-***одлучив*** *ако је* RM-*израчунљива његова карактеристична функција* $\chi_A : \mathbb{N}^k \to \{0, 1\}$,

$$\chi_A(n_1, \ldots, n_k) = \begin{cases} 1, & (n_1, \ldots, n_k) \in A, \\ 0, & (n_1, \ldots, n_k) \notin A. \end{cases}$$

Пример 21. Множење $\cdot : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ је RM-израчунљива функција. RM-програм који израчунава производ бројева, можемо добити прилагођавањем програма из примера 20 за алфабет $\{1\}$. □



Задаци

10. Доказати да је RM-израчунљива функција $f : \mathbb{N} \to \mathbb{N}$ дата са:

а) $f(x) = 2 \cdot x$; б) $f(x) = x^2$;

в) $f(x) = \begin{cases} 2, & x = 3, \\ 1, & x \neq 3; \end{cases}$ г) $f(x) = \begin{cases} 2, & x = 3 \text{ или } x = 5, \\ 1, & \text{иначе}; \end{cases}$

11. Доказати да је RM-израчунљива функција $f : \mathbb{N}^2 \to \mathbb{N}$ дата са:

а) $f(x, y) = \min(x, y) = \begin{cases} x, & x \leqslant y, \\ y, & x > y; \end{cases}$

б) $f(x, y) = x \cdot_2 y = \mathrm{rm}(2, xy)$.

12. Доказати да су RM-одлучиве бинарне релације *једнакост* $(=)$, *уређење* $(\leqslant)$ и *дељивост* $(|)$ скупа $\mathbb{N}$.

13. Саставити RM-програм if $R_i = R_j$? then $\mathbb{P}_1$ else $\mathbb{P}_2$.

Однос Тјурингових машина и регистар машина

Сваки RM-програм може се симулирати Тјуринговом машином.

Нека је $\mathbb{P} = (I_1, \ldots, I_n)$ произвољан RM-програм над алфабетом $\Sigma = \{\mathsf{s}_1, \ldots, \mathsf{s}_m\}$. Претпоставимо да је $\mathbb{P}$ у стандардној форми. Конструисаћемо Тјурингову машину $\mathbb{T}$, са улазним алфабетом Σ, алфабетом траке $\Gamma = \Sigma \cup \{1, \sqcup\}$ (при чему $1, \sqcup \notin \Sigma$), почетним стањем q_0 и једним завршним стањем q_{stop} такву да за све $k \geqslant 0$ и све речи $w_1, \ldots, w_k \in \Sigma^*$, важе следећи услови:

(1) ако $\mathbb{P}(w_1, \ldots, w_k) \uparrow$, онда $\mathbb{T}(w_1, \ldots, w_k) \uparrow$,

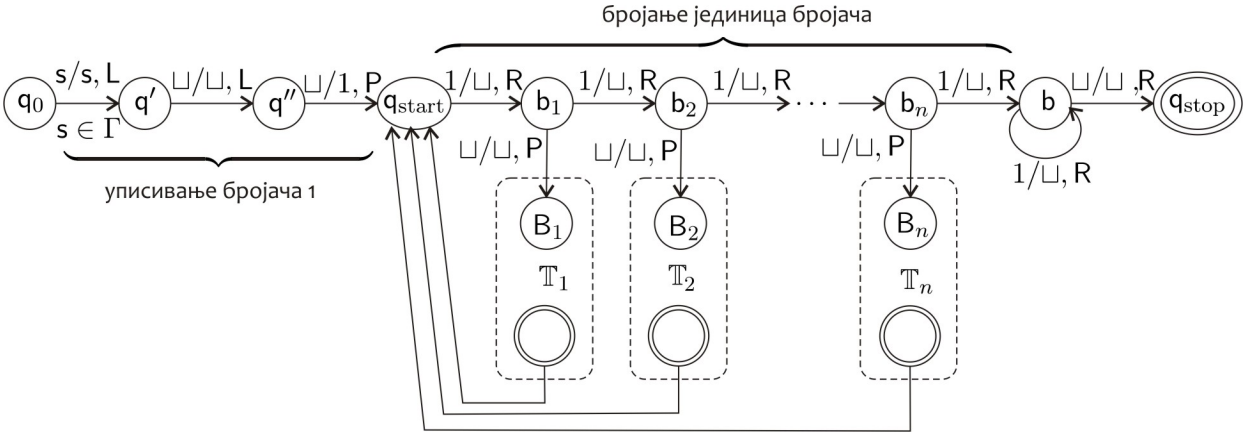
(2) ако $\mathbb{P}(w_1, \ldots, w_k) \downarrow$, онда $\mathbb{T}(w_1, \ldots, w_k) \downarrow$; штавише ако

$$1; w_1, \ldots, w_k \to^*_{\mathbb{P}} n+1; v_1, \ldots, v_d, \text{ где је } d = \max\{k, \|\mathbb{P}\|\},$$

онда $\mathsf{q}_0 w_1 \sqcup w_2 \sqcup \cdots \sqcup w_k \to^*_{\mathbb{T}} \mathsf{q}_{\text{stop}} v_1 \sqcup \cdots \sqcup v_d$.

Интуитивно је јасно да се за сваки RM-програм може конструисати Тјурингова машина $\mathbb{T}$ која задовољава услове (1) и (2). У наставку наводимо детаље поменуте конструкције. Иначе, сама конструкција неће бити од великог значаја у осталим поглављима, тако да нема потребе да се трајно памти и зато је наводимо ситнијим словима.

Конструкција машине $\mathbb{T}$. Први задатак машине $\mathbb{T}$ је да почетну конфигурацију $\mathsf{q}_0 w_1 \sqcup w_2 \sqcup \cdots \sqcup w_k$ преради у $\mathsf{q}_{\text{start}} 1 \sqcup w_1 \sqcup w_2 \sqcup \cdots \sqcup w_k$, а затим симулира инструкцију I_1. Наравно, $\mathbb{T}$ треба да садржи подмашине $\mathbb{T}_i$ за симулацију сваке од инструкција I_i, $1 \leqslant i \leqslant n$. Када $\mathbb{T}$ достигне конфигурацију облика $\mathsf{q}_{\text{start}} 1^i \sqcup u_1 \sqcup u_2 \sqcup \cdots \sqcup u_d$, $1 \leqslant i \leqslant n$, треба да „преброји“ јединице *бројача* 1^i и претходну конфигурацију преради у $\mathsf{B}_i \sqcup u_1 \sqcup \ldots \sqcup u_d$. Стање B_i је почетно стање подмашине $\mathbb{T}_i$ која обавља следећи задатак $\mathsf{B}_i \sqcup u_1 \sqcup \ldots \sqcup u_d \to^*_{\mathbb{T}_i} \mathsf{q}_{\text{start}} 1^j \sqcup u'_1 \sqcup \cdots \sqcup u'_d$, при чему је $\mathsf{NEXT}_{\mathbb{P}}(i; u_1, \ldots, u_d) = (j; u'_1, \ldots, u'_d)$. Када $\mathbb{T}$ достигне конфигурацију $\mathsf{q}_{\text{start}} 1^{n+1} \sqcup v_1 \sqcup v_2 \sqcup \cdots \sqcup v_d$ прерађује је у $\mathsf{q}_{\text{stop}} v_1 \sqcup v_2 \sqcup \cdots \sqcup v_d$. Машина $\mathbb{T}$ приказана је на следећој слици.



Преостаје још да детаљније опишемо машине $\mathbb{T}_i$. Ове машине обављају симулацију инструкције I_i у две фазе:

(1) најпре извршава акцију на одговарајућем регистру онако како налаже I_i и враћа главу на леви крај памтећи стањем каква измена је извршена;

– ако је $i.\,\mathsf{R}_k^{+\mathsf{s}} \mid \ell$ ова фаза се завршава конфигурацијом

$$\mathsf{B}_i^{+} \sqcup u_1 \sqcup \ldots \sqcup u_k\mathsf{s} \sqcup \ldots \sqcup u_d;$$

– ако је[69] $i.\,\mathsf{R}_k^{-\mathsf{s}} \mid \varepsilon\,\ell_0, \mathsf{s}_1\,\ell_1, \ldots, \mathsf{s}_m\,\ell_m$ у случају $u_k \neq \varepsilon$ ова фаза се завршава конфигурацијом

$$\mathsf{B}_i^{-\mathsf{s}} \sqcup u_1 \sqcup \ldots \sqcup u_k^{-} \sqcup \ldots \sqcup u_d,$$

при чему су $\mathsf{s} \in \Sigma$ и $u_k^{-} \in \Sigma^*$ такви да је $u_k^{-}\mathsf{s} = u_k$, а у случају $u_k = \varepsilon$ ова фаза се завршава конфигурацијом

$$\mathsf{B}_i^{-} \sqcup u_1 \sqcup \ldots \sqcup u_d,$$

без измена „регистара“;

(2) затим (здесна налево) уписује одговарајући бројач:

– ако је $i.\,R_k^{+\mathsf{s}} \mid \ell$, онда

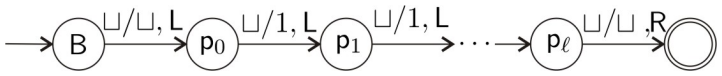
$\mathsf{B}_i^{+} \sqcup u_1 \sqcup \ldots \sqcup u_k\mathsf{s} \sqcup \ldots \sqcup u_d$ прерађује у $\mathsf{q}_{\text{start}}1^{\ell} \sqcup u_1 \sqcup \ldots \sqcup u_k\mathsf{s} \sqcup \ldots \sqcup u_d$;

– ако је $i.\,\mathsf{R}_k^{-\mathsf{s}} \mid \varepsilon\,\ell_0, \mathsf{s}_1\,\ell_1, \ldots, \mathsf{s}_m\,\ell_m$, онда

$\mathsf{B}_i^{-\mathsf{s}_t} \sqcup u_1 \sqcup \ldots \sqcup u_k^{-} \sqcup \ldots \sqcup u_d$ прерађује у $\mathsf{q}_{\text{start}}1^{\ell_t} \sqcup u_1 \sqcup \ldots \sqcup u_k^{-} \sqcup \ldots \sqcup u_d$,

$\mathsf{B}_i^{-} \sqcup u_1 \sqcup \ldots \sqcup u_d$ прерађује у $\mathsf{q}_{\text{start}}1^{\ell_0} \sqcup u_1 \sqcup \ldots \sqcup u_d$.

Фаза (2) је једноставнија. Уписивање ℓ јединица (здесна налево) обавља машина π_ℓ приказана на наредној слици.



Наредбе које ће обављати фазу (1), тј. мењају садржај одговарајућег „регистра“, састављамо надовезивањем неких једноставних Тјурингових машина. Најједноставније међу њима су [70]:

τ_{s}: $\mathsf{e}_0\mathsf{s}'\mathsf{s}\mathsf{Pe}_1$, $\mathsf{s}' \in \Gamma$

ρ: $\mathsf{e}_0\mathsf{ss}\mathsf{Re}_1$, $\mathsf{s} \in \Gamma$ $\qquad$ λ: $\mathsf{e}_0\mathsf{ss}\mathsf{Le}_1$, $\mathsf{s} \in \Gamma$

ρ^*: $\mathsf{e}_0 \sqcup \sqcup\mathsf{Re}$, $\mathsf{e}_0\mathsf{ss}\mathsf{Re}$, $\mathsf{ess}\mathsf{Re}$, $\mathsf{e} \sqcup \sqcup\mathsf{Pe}_1$, $\mathsf{s} \in \Gamma$

λ^*: $\mathsf{e}_0 \sqcup \sqcup\mathsf{Le}$, $\mathsf{e}_0\mathsf{ss}\mathsf{Le}$, $\mathsf{ess}\mathsf{Le}$, $\mathsf{e} \sqcup \sqcup\mathsf{Pe}_1$, $\mathsf{s} \in \Gamma$

Приметимо да ρ^{*k} (одн. λ^{*k}) помера главу на k-ти по реду бланко симбол здесна (одн. слева) у односу на ћелију коју на почетку скенира и зауставља се. Програм ρ^{*0} (одн. λ^{*0}) је празан програм, тј. програм без наредби.

Инструкцију I_i симулирамо на следећи начин:

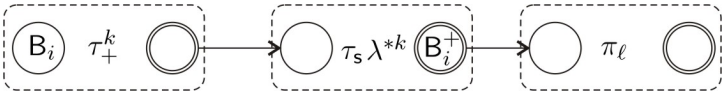
[69] $i.\,\mathsf{R}_k^{-\mathsf{s}} \mid \varepsilon\,\ell_0, \mathsf{s}_1\,\ell_1, \ldots, \mathsf{s}_m\,\ell_m$ је једноставнији запис инструкције

$i.$	R_k^{-}	
	ε	ℓ_0
	s_1	ℓ_1
	$\vdots$	
	s_m	ℓ_m

[70] τ_{s}, за $\mathsf{s} \in \Sigma$, садржај ћелије коју скенира мења у s и зауставља се у стању e_1; ρ помера главу за једно место удесно и зауставља се у стању e_1; λ помера главу за једно место улево и зауставља се у стању e_1; ρ^* (одн. λ^*) помера главу на први бланко симбол здесна (одн. слева) у односу на ћелију коју на почетку скенира, не рачунајући ту ћелију, и зауставља се у стању e_1

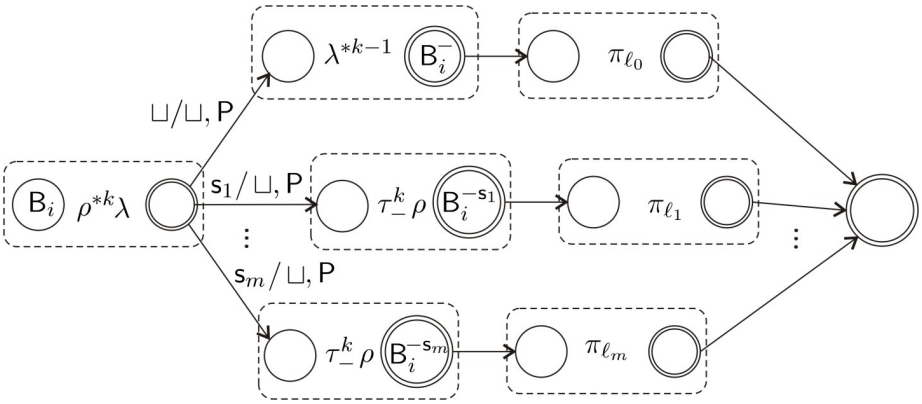
- Ако је $i.\,R_k^{+\mathsf{s}} \to \ell$, најпре иза речи u_k треба уметнути $\sqcup$, што постижемо програмом τ_+^k, при чему τ_+ садржи следеће наредбе[71]: $\mathsf{e}_0\,\sqcup\,\sqcup\mathsf{Re}$, $\mathsf{es}\,\sqcup\,\mathsf{Le}_\mathsf{s}$, $\mathsf{e}_\mathsf{s}\,\sqcup\,\mathsf{s}\mathsf{R}\bar{\mathsf{e}}$, $\bar{\mathsf{e}}\,\sqcup\,\sqcup\mathsf{Re}$, $\mathsf{e}\,\sqcup\,\sqcup\mathsf{Le}_{\mathrm{stop}}$. На τ_+^k се затим надовезује $\tau_\mathsf{s}\lambda^{*k}$ и завршава се у стању B_i^+.

Машина $\mathbb{T}_i$ која симулира $i.\,R_k^{+\mathsf{s}} \to \ell$ приказана је на наредној слици. Свака од приказаних стрелица замењује стрелице означене $\mathsf{s}/\mathsf{s}, \mathsf{P}$, $\mathsf{s} \in \Gamma$.



- Ако је $i.\,\mathsf{R}_k^{-\mathsf{s}} \mid \varepsilon\,\ell_0, \mathsf{s}_1\,\ell_1, \ldots, \mathsf{s}_m\,\ell_m$, након извршавања програма $\rho^{*k}\lambda$, глава чита последњи симбол речи u_k уколико је $u_k \neq \varepsilon$, односно $\sqcup$ уколико је $u_k = \varepsilon$.
  - Ако глава чита неки $\mathsf{s} \in \Sigma$, онда треба га обрисати (уписати $\sqcup$) без померања главе и извршити програм τ_-^k, при чему τ_- садржи следеће наредбе[72]: $\mathsf{e}_0\,\sqcup\,\sqcup\mathsf{Le}$, $\mathsf{es}\,\sqcup\,\mathsf{Re}_\mathsf{s}$, $\mathsf{e}_\mathsf{s}\,\sqcup\,\mathsf{s}\bar{\mathsf{e}}$, $\bar{\mathsf{e}}\,\sqcup\,\sqcup\mathsf{Le}$, $\mathsf{e}\,\sqcup\,\sqcup\mathsf{Pe}_{\mathrm{stop}}$. На τ_-^k се затим надовезује ρ и завршава се у стању $\mathsf{B}_i^{-\mathsf{s}}$.
  - Ако глава чита $\sqcup$, онда треба извршити наредбе програма λ^{*k-1} и завршити у стању B_i^-.

Машина $\mathbb{T}_i$ која симулира $i.\,\mathsf{R}_k^{-\mathsf{s}} \mid \varepsilon\,\ell_0, \mathsf{s}_1\,\ell_1, \ldots, \mathsf{s}_m\,\ell_m$ приказана је на наредној слици.



Свака Тјурингова машина се може симулирати RM-програмом.

Показаћемо да се уз погодно кодирање, за сваку Тјурингову машину $\mathbb{T} = (Q, \mathsf{q}_0, F, \Sigma, \sqcup, \Gamma, \tau)$ може написати RM-програм над унарним алфабетом $\{1\}$ који симулира рад машине $\mathbb{T}$.

Нека су сва стања машине $\mathbb{T}$ означена $\mathsf{q}_0, \ldots, \mathsf{q}_k, \mathsf{q}_{k+1}, \ldots, \mathsf{q}_n$, при чему је q_0 почетно стање, $\mathsf{q}_{k+1}, \ldots, \mathsf{q}_n$ су завршна стања, уз претпоставку (без губљења општости) да почетно стање није и завршно. Поређајмо у низ и све симболе траке $\mathsf{s}_0, \mathsf{s}_1, \ldots, \mathsf{s}_m$, при чему је s_0 бланко знак $\sqcup$. Сваком стању и сваком симболу придружићемо, као кôд, број у индексу:[73] $[\mathsf{q}_i] = i$ и $[\mathsf{s}_i] = i$.

[71] Програм τ_+ конфигурације облика:

$$u \quad \sqcup \quad \overset{\nabla}{\sqcup} \quad \mathsf{s}_{i_1} \quad \cdots \quad \mathsf{s}_{i_t} \quad \sqcup \quad v$$

прерађује у:

$$u \quad \sqcup \quad \mathsf{s}_{i_1} \quad \cdots \quad \overset{\nabla}{\mathsf{s}_{i_t}} \quad \sqcup \quad \sqcup \quad v$$

[72] Програм τ_- конфигурацију облика:

$$u \quad \sqcup \quad \mathsf{s}_{i_1} \quad \cdots \quad \overset{\nabla}{\mathsf{s}_{i_t}} \quad \sqcup \quad \sqcup \quad v$$

прерађује у

$$\overset{\nabla}{u} \quad \sqcup \quad \sqcup \quad \mathsf{s}_{i_1} \quad \cdots \quad \mathsf{s}_{i_t} \quad \sqcup \quad v$$

[73] Једнакост $[\mathsf{q}_i] = i$ читамо „кôд стања q_i једнак је броју i“.

Знацима R, P, L редом придружујемо бројеве 0, 1, 2: $[\mathsf{R}] = 0$, $[\mathsf{P}] = 1$, $[\mathsf{L}] = 2$.

Сваку конфигурацију машине $\mathbb{T}$, тј. реч из $\Gamma^* Q \Gamma^+$,

$$\mathsf{C}:\ u_\ell \cdots u_1 \mathsf{q}_i v_1 \cdots v_r,\ \ell \geqslant 0, r > 0, u_j, v_k \in \Gamma,$$

кодирамо бројем $[\mathsf{C}] = \lceil i, \lceil [u_1], \ldots, [u_\ell] \rceil, \lceil [v_1], \ldots, [v_r] \rceil \rceil$. При томе, подразумеваћемо да са леве и десне стране речи која описује конфигурацију може бити уписано коначно много бланко знакова.

Ако је $\overline{w}$ неки коначан низ речи улазног алфабета машине $\mathbb{T}$, почетну конфигурацију одређену овим улазом означићемо $\mathsf{C}_{\overline{w}}$.

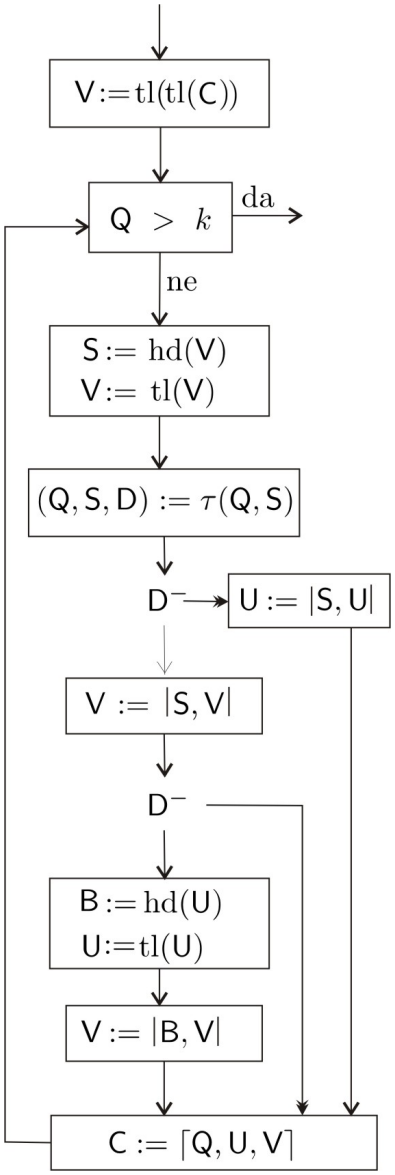
Саставићемо RM-програм такав да за све $k \geqslant 0$ и све речи $w_1, \ldots, w_k \in \Sigma^*$, важе следећи услови:

(1) ако $\mathbb{T}(w_1, \ldots, w_k) \uparrow$, онда $\mathbb{P}([\mathsf{C}_{\overline{w}}]) \uparrow$,

(2) ако $\mathbb{T}(w_1, \ldots, w_k) \downarrow$, онда $\mathbb{P}([\mathsf{C}_{\overline{w}}]) \downarrow$; штавише уколико $\mathsf{C}_{\overline{w}} \to^*_{\mathbb{T}} \mathsf{C}_{\text{stop}}$, онда је $[\mathsf{C}_{\text{stop}}]$ садржај првог регистра у завршној конфигурацији израчунавања програма $\mathbb{P}$ за улаз $[\mathsf{C}_{\overline{w}}]$.

Програм $\mathbb{P}$ приказан је на слици десно. Због прегледности, за важне регистре користимо следеће сугестивне ознаке: C – ознака за регистар R_1 у коме се чувају кôдови конфигурација израчунавања машине $\mathbb{T}$ за улаз $\overline{w}$; U – за кôд речи лево од ћелије која се чита, без те ћелије; V – за кôд речи десно од ћелије која се чита, са том ћелијом; Q – за кôд текућег стања; S – за кôд текућег симбола (који чита глава); D – за кôд „помераја“ главе (вредности ових регистара могу бити $[\mathsf{R}] = 0$, $[\mathsf{P}] = 1$, $[\mathsf{L}] = 2$); B – помоћни регистар.

За сваку петорку $\mathsf{qss'}D\mathsf{q'}$ програма τ, јасно је да се може саставити RM-програм $(\mathsf{Q}, \mathsf{S}, \mathsf{D}) := \tau(\mathsf{Q}, \mathsf{S})$ који обавља следећи задатак: ако су садржаји регистара Q и S редом једнаки $[\mathsf{q}]$ и $[\mathsf{s}]$, онда у регистре Q, S, D редом упиши $[\mathsf{q'}]$, $[\mathsf{s'}]$ и $[D]$.



Приказани граф можемо замишљати и као приказ *механизма* који извршава Тјурингове програме. Програмер би користио овај механизам тако што би писао програм $(\mathsf{Q}, \mathsf{S}, \mathsf{D}) := \tau(\mathsf{Q}, \mathsf{S})$, прецизирако параметар k, а улазе за свој програм сопштавао у облику $\lceil 0, 0, \lceil \cdots \rceil \rceil$. Механизам ће завршну конфигурацију написаног Тјурингог програма вратити у облику $\lceil \mathsf{Q}, \lceil \cdots \rceil, \lceil \cdots \rceil \rceil$, који само треба декодирати.

Пример 22. Нека је $\mathbb{T} = (\{\mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2\}, \mathsf{q}_0, \{\mathsf{q}_2\}, \{\sqcup, 1\}, \sqcup, \{1\}, \tau)$, где је τ одређено наредбама $\mathsf{q}_0 \sqcup \sqcup \mathsf{L}\mathsf{q}_2$, $\mathsf{q}_1 \sqcup 1\mathsf{L}\mathsf{q}_1$, $\mathsf{q}_0 1 \sqcup \mathsf{R}\mathsf{q}_1$, $\mathsf{q}_1 1 \sqcup \mathsf{R}\mathsf{q}_0$, Тјурингова машина из Примера 4. Илустроваћемо рад RM-програм $\mathbb{P}$, над $\{1\}$, који симулира $\mathbb{T}$, тако што ћемо га приказати упоредо са одговарајућим израчунавањем машине $\mathbb{T}$ за улаз 11. Пре тога, истичемо да је кодирање машине $\mathbb{T}$ обављено по „кључу“:

$$\begin{pmatrix} \mathsf{q}_0 & \mathsf{q}_1 & \mathsf{q}_2 & \sqcup & 1 & \mathsf{R} & \mathsf{P} & \mathsf{L} \\ 0 & 1 & 2 & 0 & 1 & 0 & 1 & 2 \end{pmatrix}$$

Кôд почетне конфигурације машине $\mathbb{T}$ за улаз 11 је једнак $\lceil 0, 0, \lceil 1, 1 \rceil \rceil$. Зато приказујемо израчунавање програма $\mathbb{P}$ за почетну конфигурацију у којој је садржај регистра C једнак $\lceil 0, 0, \lceil 1, 1 \rceil \rceil$, а садржај сваког другог регистара је 0.

q_0
∇ $\quad C := \lceil 0, 0, \lceil [1], [1] \rceil \rceil$
$\cdots \quad \sqcup \quad 1 \quad 1 \quad \sqcup \quad \cdots$

$\downarrow \mathsf{q}_0 1 \sqcup \mathsf{R} \mathsf{q}_1$ | $V := \mathrm{tl}(\mathrm{tl}(C)) = \lceil [1], [1] \rceil$
Није $\mathsf{Q} > 1$, па је $S := \mathrm{hd}(V) = [1]$, $V := \mathrm{tl}(V) = \lceil [1] \rceil$.
Како је $\mathsf{Q} = 0$, $\mathsf{S} = [1]$ и $\tau(\mathsf{q}_0, 1) = (\sqcup, \mathsf{R}, \mathsf{q}_1)$
након извршавања $(\mathsf{Q}, \mathsf{S}, \mathsf{D}) := \tau(\mathsf{Q}, \mathsf{S})$ биће
$\mathsf{Q} = 1$, $\mathsf{S} = [\sqcup] = 0$, $\mathsf{D} = [\mathsf{R}] = 0$.
Из $\mathsf{D} = 0$ следи $\mathsf{U} := |\mathsf{S}, \mathsf{U}| = \lceil [\sqcup], [\sqcup] \rceil$ и

q_1
∇ $\quad \mathsf{C} := \lceil \mathsf{Q}, \mathsf{U}, \mathsf{V} \rceil = \lceil 1, \lceil [\sqcup], [\sqcup] \rceil, \lceil [1] \rceil \rceil$.
$\cdots \quad \sqcup \quad \sqcup \quad 1 \quad \sqcup \quad \cdots$

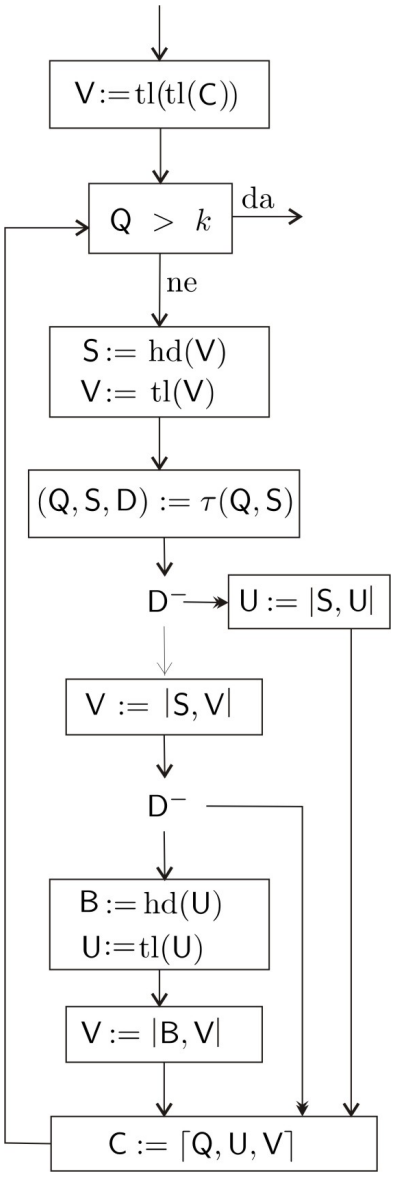
$\downarrow \mathsf{q}_1 1 \sqcup \mathsf{R} \mathsf{q}_0$ | $V := \mathrm{tl}(\mathrm{tl}(C)) = \lceil [1] \rceil$
Није $\mathsf{Q} > 1$, па је $S := \mathrm{hd}(V) = [1]$, $V := \mathrm{tl}(V) = 0 = [\sqcup]$.
Како је $\mathsf{Q} = 1$, $\mathsf{S} = [1]$ и $\tau(\mathsf{q}_1, 1) = (\sqcup, \mathsf{R}, \mathsf{q}_0)$
након извршавања $(\mathsf{Q}, \mathsf{S}, \mathsf{D}) := \tau(\mathsf{Q}, \mathsf{S})$ биће
$\mathsf{Q} = 0$, $\mathsf{S} = [\sqcup] = 0$, $\mathsf{D} = [\mathsf{R}] = 0$.
Из $\mathsf{D} = 0$ следи $\mathsf{U} := |\mathsf{S}, \mathsf{U}| = \lceil [\sqcup], [\sqcup], [\sqcup] \rceil$ и

q_0
∇ $\quad \mathsf{C} : \lceil \mathsf{Q}, \mathsf{U}, \mathsf{V} \rceil = \lceil 1, \lceil [\sqcup], [\sqcup], [\sqcup] \rceil, [\sqcup] \rceil$.
$\cdots \quad \sqcup \quad \sqcup \quad \sqcup \quad \sqcup \quad \cdots$

$\downarrow \mathsf{q}_0 \sqcup \sqcup \mathsf{L} \mathsf{q}_2$ | $V := \mathrm{tl}(\mathrm{tl}(C)) = \lceil [\sqcup] \rceil$
Није $\mathsf{Q} > 1$, па је $S := \mathrm{hd}(V) = [\sqcup]$, $V := \mathrm{tl}(V) = 0 = [\sqcup]$.
Како је $\mathsf{Q} = 0$, $\mathsf{S} = [\sqcup]$ и $\tau(\mathsf{q}_0, \sqcup) = (\sqcup, \mathsf{L}, \mathsf{q}_2)$
након извршавања $(\mathsf{Q}, \mathsf{S}, \mathsf{D}) := \tau(\mathsf{Q}, \mathsf{S})$ биће
$\mathsf{Q} = 2$, $\mathsf{S} = [\sqcup]$, $\mathsf{D} = [\mathsf{L}] = 1$.
Из $\mathsf{D} = 1$ следи $\mathsf{D} := \mathsf{D} - 1 = 0$, $\mathsf{V} := |\mathsf{S}, \mathsf{V}| = \lceil [\sqcup], [\sqcup] \rceil$ и

q_2
∇ $\quad \mathsf{C} := \lceil \mathsf{Q}, \mathsf{U}, \mathsf{V} \rceil = \lceil 2, \lceil [\sqcup], [\sqcup], [\sqcup] \rceil, \lceil [\sqcup], [\sqcup] \rceil \rceil$.
$\cdots \quad \sqcup \quad \sqcup \quad \sqcup \quad \sqcup \quad \cdots$

STOP | Како јесте $\mathsf{Q} = 2 > 1$, програм $\mathbb{P}$ се зауставља.



Издвајамо једну очигледну последицу претходних разматрања.

Последица 1. *За сваку функцију* $f : \Sigma^{*k} \to \Sigma^*$ *важи:*

f *је* RM*-израчунљива акко је* TM*-израчунљива.*

Чињеница да се све Тјурингове машине, а самим тим и сви RM-програми, могу симулирати RM-програмима над унарним алфабетом омогућава нам да у теоријским разматрањима потпуну пажњу усмеримо на израчунљивост функција над скупом $\mathbb{N}$, односно на одлучивост релација над $\mathbb{N}$.

Парцијалне RM-израчунљиве функције

Основни циљ у неколико наредних одељака је да што јасније сагледамо и разумемо шта је све могуће израчунавати RM-програмима (над унарним алфабетом). Приметимо најпре да сваки RM-програм $\mathbb{P}$ за свако $k \geqslant 1$ одређује један подскуп од $\mathbb{N}^k$ који садржи све оне k-торке природних бројева за које се $\mathbb{P}$ зауставља:

$$W_{\mathbb{P}}^{(k)} = \{(x_1, \ldots, x_k) \in \mathbb{N}^k \mid \mathbb{P}(x_1, \ldots, x_k) \downarrow\}.$$

Наравно, постоје програми $\mathbb{P}$ такви да је $W_{\mathbb{P}}^{(k)} = \mathbb{N}^k$, али и они да је $W_{\mathbb{P}}^{(k)} \neq \mathbb{N}^k$; штавише постоје и такви да је $W_{\mathbb{P}}^{(k)} = \emptyset$.

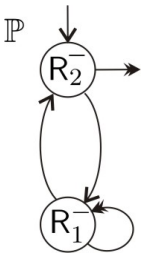
Према ранијем договору, уколико $\mathbb{P}(x_1, \ldots, x_k) \downarrow$, за резултат израчунавања програма $\mathbb{P}$ за улаз $\vec{x} = (x_1, \ldots, x_k)$ узимамо садржај регистра R_1 у завршној конфигурацији. Дакле, програм $\mathbb{P}$, за свако $k \geqslant 1$ одређује јединствену функцију $\varphi_{\mathbb{P}}^{(k)} : W_{\mathbb{P}}^{(k)} \to \mathbb{N}$,

$$\varphi_{\mathbb{P}}^{(k)}(\vec{x}) = \text{садржају регистра } R_1 \text{ у завршној конфигурацији израчунавања програма } \mathbb{P} \text{ за улаз } \vec{x}.$$

Када посматрамо унарну функцију коју израчунава неки програм $\mathbb{P}$, уместо $\varphi_{\mathbb{P}}^{(1)}$ писаћемо само $\varphi_{\mathbb{P}}$; другим речима, ако не наведемо дужину функције коју израчунава $\mathbb{P}$, подразумеваћемо да је реч о унарној функцији.

Пример 23. Програм $\mathbb{P}$: $1.\mathsf{R}_2^- \mid 3, 2 \;\; 2.\mathsf{R}_1^- \mid 2, 1$, за $k \geqslant 2$, израчунава парцијалну функцију (парцијално одузимање)

$$\varphi_{\mathbb{P}}^{(k)}(x_1, x_2, \ldots, x_k) = \begin{cases} x_1 - x_2, & x_1 \geqslant x_2, \\ \uparrow, & x_1 < x_2. \end{cases}$$



Исти програм израчунава и тоталну (унарну) функцију $\varphi_{\mathbb{P}}(x) = x$ (идентичко пресликавање).

Узимајући у обзир врсту функција које одређују RM-програми, пажњу усмеравамо на тзв. парцијалне функције и на њих проширујемо појам RM-израчунљивости.

Функцију $\varphi : D \to \mathbb{N}$, где је $D \subseteq \mathbb{N}^k$, називамо k-арном **парцијалном функцијом**[78]; када је $D = \mathbb{N}^k$ кажемо да је φ **тотална функција**.[79] Посебно, и празан скуп сматрамо парцијалном функцијом и називамо је *празна функција*. Домен празне функције је празан скуп, и како је $\emptyset \subseteq \mathbb{N}^k$, за свако $k \geqslant 1$, празну функцију можемо замишљати као функцију било које дужине.

[78] 1-арне функције називамо унарним; 2-арне бинарним, 3-арне тернарним.

[79] Тоталне функције су специјални случајеви парцијалних функција.

Ако је φ k-арна парцијална функција, њен домен означавамо $\mathrm{dom}(\varphi)$; дакле, $\mathrm{dom}(\varphi) \subseteq \mathbb{N}^k$. Уместо $\vec{x} \in \mathrm{dom}(\varphi)$ пишемо $\varphi(\vec{x}) \downarrow$, а уместо $\vec{x} \notin \mathrm{dom}(\varphi)$ пишемо $\varphi(\vec{x}) \uparrow$. Скуп вредности функције φ означавамо $\mathrm{ran}(\varphi)$; дакле, $\mathrm{ran}(\varphi) \subseteq \mathbb{N}$. Уместо $\varphi(\vec{x}) \downarrow \wedge \varphi(\vec{x}) = y$ краће пишемо $\varphi(\vec{x}) \downarrow y$

Ако су φ и ψ две парцијалне функције исте дужине k, за произвољно $\vec{x} \in \mathbb{N}^k$, формула $\varphi(\vec{x}) \simeq \psi(\vec{x})$ значи:

- или обе вредности $\varphi(\vec{x})$ и $\psi(\vec{x})$ нису дефинисане (тј. $\varphi(\vec{x}) \uparrow$ и $\psi(\vec{x}) \uparrow$)
- или су обе вредности $\varphi(\vec{x})$ и $\psi(\vec{x})$ дефинисане и једнаке (тј. $\varphi(\vec{x}) \downarrow, \psi(\vec{x}) \downarrow$ и $\varphi(\vec{x}) = \psi(\vec{x})$).

Једнаке могу бити само две парцијалне функције исте дужине[80]:

$$\varphi = \psi \overset{\text{def}}{\Leftrightarrow} \forall \vec{x}\,(\varphi(\vec{x}) \simeq \psi(\vec{x})).$$

[80] Функције φ и ψ су једнаке ако имају исти домен и једнаке вредности за све аргументе (из заједничког) домена.

Дефиниција 5. *Парцијална k-арна функција φ је* RM-*израчунљива ако постоји* RM-*програм $\mathbb{P}$ такав да за све $\vec{x} \in \mathbb{N}^k$ важи $\varphi(\vec{x}) \simeq \varphi_{\mathbb{P}}^{(k)}(\vec{x})$, односно:*

- *ако $\vec{x} \in \mathrm{dom}(\varphi)$, онда се извршавање програма $\mathbb{P}$ за улаз $\vec{x}$ зауставља и садржај регистра R_1 у завршној конфигурацији је $\varphi(\vec{x})$;*
- *ако $\vec{x} \notin \mathrm{dom}(\varphi)$, онда се извршавање програма $\mathbb{P}$ за улаз $\vec{x}$ не зауставља.*

Лема 1. *Основне функције:*

- *Нула-функција* $\mathbf{0} : \mathbb{N} \to \mathbb{N}, \mathbf{0}(x) = 0$,
- *Следбеник* $s : \mathbb{N} \to \mathbb{N}, s(x) = x + 1$,
- *Пројекције* $\Pi_i^k : \mathbb{N}^k \to \mathbb{N}, \Pi_i^k(x_1, \ldots, x_k) = x_i$ $(1 \leqslant i \leqslant k)$,

јесу RM-*израчунљиве.*

Доказ. Нула-функција $\mathbf{0} : \mathbb{N} \to \mathbb{N}, \mathbf{0}(x) = 0$, јесте RM-израчунљива; израчунава је програм $R_1 := \varepsilon$ (видети пример 17).
Следбеник $s : \mathbb{N} \to \mathbb{N}, s(x) = x + 1$, јесте RM-израчунљива; израчунава је програм: 1. $R_1^+ \mid 2$.
Пројекције $\Pi_i^k : \mathbb{N}^k \to \mathbb{N}, \Pi_i^k(x_1, \ldots, x_k) = x_i$ $(1 \leqslant i \leqslant k)$ јесу RM-израчунљиве. Ако је $i > 1$, функцију Π_i^k израчунава програм $R_1 := R_i$ (видети пример 17). Ако је $i = 1$, функцију Π_1^k израчунава (на пример) програм: 1. $R_1^+ \mid 2 \qquad 2. \quad R_1^- \mid 3, 3.$ □

Најзачајнију улогу у карактеризацији скупа свих RM-израчунљивих функција имају општи принципи дефинисања функција које издвајамо у наставку, а за које је затворен поменути скуп функција.

Супституција

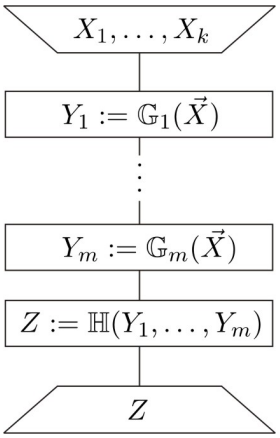
Први принцип који наводимо је супституција.[81] Ако су $g_1, \ldots, g_m$ неке k-арне парцијалне функције и h нека m-арна парцијална функција, онда за k-арну функцију f дефинисану са:

$$\text{(Sup)} \qquad f(\vec{x}) \simeq h(g_1(\vec{x}), \ldots, g_m(\vec{x})),\ \vec{x} \in \mathbb{N}^k$$

[81] Уопштено говорећи, супституција је подразумевани принцип који примењујемо приликом најједноставније методе дефинисања функција – помоћу израза. Полазећи од променљивих и симбола константи, изразе градимо тако што на већ изграђене $t_1, \ldots, t_k$ примењујемо неку k-арну функцију F и градимо нови израз $F(t_1, \ldots, t_k)$.

кажемо да је добијена **супституцијом** функција $g_1, \ldots, g_m$ у h и пишемо $f = \mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$.

Показаћемо да је скуп RM-израчунљивих функција затворен за супституцију: ако су $g_1, \ldots, g_m$ неке k-арне RM-израчунљиве функције, које израчунавају редом програми $\mathbb{G}_1, \ldots, \mathbb{G}_m$, и h је нека m-арна RM-израчунљива функција, коју израчунава програм $\mathbb{H}$, онда се може саставити програм $\mathbb{F}$ који за улаз $\vec{x} \in \mathbb{N}^k$ рачуна $h(g_1(\vec{x}), \ldots, g_m(\vec{x}))$ спроводећи следећи неформално описан поступак:



- рачуна $g_1(\vec{x})$, користећи програм $\mathbb{G}_1$; ако $\mathbb{G}_1(\vec{x}) \downarrow$, памти резултат $y_1 := g_1(\vec{x})$ и прелази на наредни корак;

$\vdots$

- рачуна $g_m(\vec{x})$, користећи програм $\mathbb{G}_m$; ако $\mathbb{G}_m(\vec{x}) \downarrow$, памти резултат $y_m := g_m(\vec{x})$ и прелази на наредни корак;

- рачуна $h(y_1, \ldots, y_m)$, користећи програм $\mathbb{H}$; ако $\mathbb{H}(y_1, \ldots, y_m) \downarrow$, зауставља се и враћа резултат $y := h(y_1, \ldots, y_m)$.

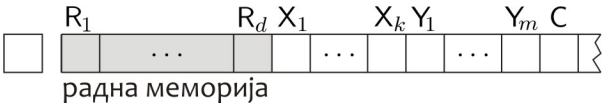
Приметимо да је могуће да се наведени поступак никада не завршава уколико се „заглавимо“ због незаустављања неког од програма $\mathbb{G}_1, \ldots, \mathbb{G}_m$ и $\mathbb{H}$ за одговарајуће улазе.

Теорема 3. *Ако су RM-израчунљиве k-арне функције $g_1, \ldots, g_m$ и m-арна функција h, онда је таква и функција $\mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$.*

Доказ. Нека су $\mathbb{G}_1, \ldots, \mathbb{G}_m$ и $\mathbb{H}$ редом програми који израчунавају k-арне функције $g_1, \ldots, g_m$ и m-арну функцију h. Нека је

$$d = \max\{k, m, ||\mathbb{G}_1||, \ldots, ||\mathbb{G}_m||, ||\mathbb{H}||\} + 1.$$

Низ регистара $R_1, \ldots, R_d$ називаћемо *радном меморијом*. Регистре $R_{d+1}, \ldots, R_{d+k}$, $R_{d+k+1}, \ldots, R_{d+k+m}$, $R_{d+k+m+1}$ редом ћемо краће означити $X_1, \ldots, X_k$, $Y_1, \ldots, Y_m$, C, при чему је C регистар који користимо за копирање (па га зато не наводимо експлицитно у наредном програму).



Програм $\mathbb{F}$ за k-арну функцију $\mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$ добијамо надовезивањем следећих програма:

$(X_1, \ldots, X_k) := (R_1, \ldots, R_k)$ означава програм који садржаје регистара $R_1, \ldots, R_k$ редом копира у $X_1, \ldots, X_k$.

$(X_1, \ldots, X_k) := (R_1, \ldots, R_k)$
$\mathbb{G}_1$
$Y_1 := R_1$
$(R_1, \ldots, R_d) := (X_1, \ldots, X_k, 0, \ldots, 0)$
$\mathbb{G}_2$
$Y_2 := R_1$
$(R_1, \ldots, R_d) := (X_1, \ldots, X_k, 0, \ldots, 0)$
$\vdots$
$\mathbb{G}_m$
$Y_m := R_1$
$(R_1, \ldots, R_d) := (Y_1, \ldots, Y_m, 0, \ldots, 0)$
$\mathbb{H}$

□

Претходна теорема нам омогућава да RM-израчунљивост неких функција покажемо без писања одговарајућих RM-програма.

ПРИМЕР 24. *Константне функције* су RM-израчунљиве. До овог закључка долазимо користећи RM-израчунљивост нула-функције (Лема 1), претходну теорему и принцип математичке индукције.

$\mathbf{1} : \mathbb{N} \to \mathbb{N}, \mathbf{1}(x) = 1$ $[\mathbf{1} = \mathrm{Sup}_1^1(s; \mathbf{0})]$
$\mathbf{2} : \mathbb{N} \to \mathbb{N}, \mathbf{2}(x) = 2$ $[\mathbf{2} = \mathrm{Sup}_1^1(s; \mathbf{1})]$

За свако $m \in \mathbb{N}$, функција $\mathbf{m} : \mathbb{N} \to \mathbb{N}, \mathbf{m}(x) = m$ је RM-израчунљива.

Слично добијамо да је за све $m \in \mathbb{N}$ и $k \in \mathbb{N}^+$ функција $\mathbf{m}_k : \mathbb{N}^k \to \mathbb{N}$, $\mathbf{m}_k(\vec{x}) = m$ такође RM-израчунљива. $[\mathbf{m}_k = \mathrm{Sup}_1^k(\mathbf{m}; \Pi_1^k)]$

ПРИМЕР 25. *Премештање и фиксирање аргумената:* ако је $h : \mathbb{N}^k \to \mathbb{N}$ RM-израчунљива функција, таква ће бити и m-арна функција h':

$$h'(x_1, \ldots, x_m) \simeq h(\underbrace{-}_{(1)}, \ldots, \underbrace{-}_{(k)}),$$

коју дефинишемо тако што на места $(1), \ldots, (k)$ уписујемо, у произвољном поретку уз могућа понављања, променљиве $x_1, \ldots, x_m$ или константе.

На пример, ако је $h : \mathbb{N}^3 \to \mathbb{N}$ нека RM-израчунљива, такве су и:

- $h_1 : \mathbb{N} \to \mathbb{N}, h_1(x_1) \stackrel{\text{def}}{=} h(x_1, 2, x_1); h_1 = \mathrm{Sup}_3^1(h; \Pi_1^1, \mathbf{2}, \Pi_1^1)$.
- $h_2 : \mathbb{N}^2 \to \mathbb{N}, h_2(x_1, x_2) \stackrel{\text{def}}{=} h(x_2, x_1, x_1); h_2 = \mathrm{Sup}_3^2(h; \Pi_2^2, \Pi_1^2, \Pi_1^2)$;
- $h_3 : \mathbb{N}^3 \to \mathbb{N}, h_3(x_1, x_2, x_3) \stackrel{\text{def}}{=} h(x_2, x_3, x_1); h_3 = \mathrm{Sup}_3^3(h; \Pi_2^3, \Pi_3^3, \Pi_1^3)$;
- $h_4 : \mathbb{N}^4 \to \mathbb{N}, h_4(x_1, x_2, x_3, x_4) \stackrel{\text{def}}{=} h(x_3, 5, x_2); h_4 = \mathrm{Sup}_3^4(h; \Pi_3^4, \mathbf{5}_4, \Pi_2^4)$.
- $h_5 : \mathbb{N}^5 \to \mathbb{N}, h_5(x_1, x_2, x_3, x_4, x_5) \stackrel{\text{def}}{=} h(x_4, 1, x_1); h_5 = \mathrm{Sup}_3^5(h; \Pi_4^5, \mathbf{1}_5, \Pi_1^5)$.

Примитивна рекурзија

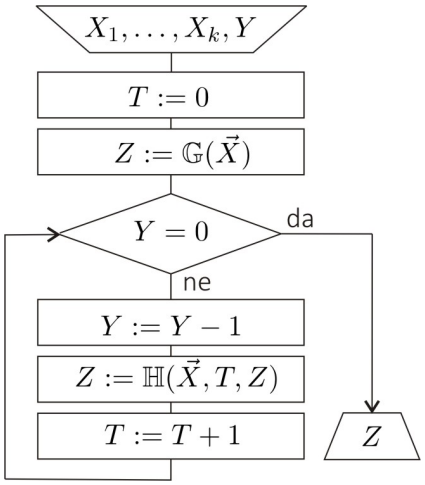
Ако је g нека k-арна парцијална функција и h нека $(k+2)$-арна парцијална функција, онда за $(k+1)$-арну функцију f дефинисану са:[84]

$$\text{(Rec)} \qquad \left| \begin{array}{l} f(\vec{x},0) \simeq g(\vec{x}), \\ f(\vec{x},y+1) \simeq h(\vec{x},y,f(\vec{x},y)), \end{array} \right.$$

кажемо да је добијена **примитивном рекурзијом** функција g и h и пишемо $f = \mathrm{Rec}^{k+1}(g,h)$. Користећи програме $\mathbb{G}$ и $\mathbb{H}$ који редом израчунавају k-арну функцију g и $(k+2)$-арну функцију h, може се саставити програм $\mathbb{F}$ који за улаз $(\vec{x},y) \in \mathbb{N}^{k+1}$ рачуна $f(\vec{x},y)$, где је f дато формулама (Rec). Програм $\mathbb{F}$ обавља следећи задатак користећи променљиву t, при чему је иницијално $t := 0$:

1. рачуна $g(\vec{x})$, користећи $\mathbb{G}$, и уколико $\mathbb{G}(\vec{x}) \downarrow$ памти резултат $z := g(\vec{x})$;
2. ако је $y = 0$ зауставља се и враћа резултат z, а ако је $y \neq 0$, смањује y за 1 ($y := y \dot{-} 1$) и прелази на наредни корак;
3. рачуна $h(\vec{x},t,z)$, користећи $\mathbb{H}$; ако $\mathbb{H}(\vec{x},t,z) \downarrow$ памти резултат $z := h(\vec{x},t,z)$, увећава t за 1 ($t := t+1$) и прелази на корак 2.

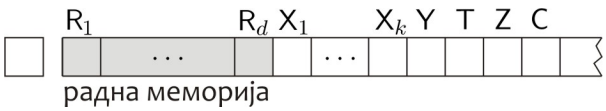
[84] Није тешко уочити да (Rec) заиста дефинише јединствену функцију f.



Теорема 4. *Ако су RM-израчунљиве k-арна функција g и $(k+2)$-арна функција h, онда је таква и функција* $\mathrm{Rec}^{k+1}(g,h)$.

Доказ. Ако програми $\mathbb{G}$ и $\mathbb{H}$ редом израчунавају k-арну функцију g и $(k+2)$-арну функцију h, нека је $d = \max\{k+2, ||\mathbb{G}||, ||\mathbb{H}||\}$. Низ регистара $R_1, \ldots, R_d$ називаћемо *радном меморијом*. Регистре $R_{d+1}, \ldots, R_{d+k}$, R_{d+k+1}, R_{d+k+2}, R_{d+k+3}, R_{d+k+4} редом ћемо краће означити $X_1, \ldots, X_k$, Y, T, Z, C, при чему је C регистар који користимо за копирање. Програм $\mathbb{F}$ за $(k+1)$-арну функцију $\mathrm{Rec}^{k+1}(g^{(k)}, h^{(k+2)})$ приказан је испод.

$(X_1, \ldots, X_k, Y) := (R_1, \ldots, R_k, R_{k+1})$
↓
$R_{k+1} := 0$
↓
$\mathbb{G}$
↓
$Z := R_1$
↓
Y^- ⟶»
↓
$(R_1, \ldots, R_d) := (X_1, \ldots, X_k, T, Z, 0, \ldots, 0)$
↓
$\mathbb{H}$
↓
$T := T + 1$



□

Користећи претходну теорему можемо доказати RM-израчунљивост великог броја функција, без писања RM-програма.

ПРИМЕР 26. *Сабирање*, *множење* и *степеновање* дефинисали смо у првом поглављу на следећи начин:

$$\text{(S)}\left|\begin{array}{l}+(x,0)=x,\\ +(x,s(y))=s(+(x,y));\end{array}\right. \quad \text{(M)}\left|\begin{array}{l}\cdot(x,0)=0,\\ \cdot(x,s(y))=+(\cdot(x,y),x);\end{array}\right. \quad \text{(E)}\left|\begin{array}{l}\exp(x,0)=1,\\ \exp(x,s(y))=\cdot(\exp(x,y),x).\end{array}\right.$$

Сваки пар једнакости (S), (M) и (E) није тешко „уклопити“ у шему примитивне рекурзије (која дефинише бинарне функције)[86], тј. у сваком од случајева пронаћи функције на које је примењен оператор Rec^2. За сабирање, реч је о функцијама Π^1_1 (идентичко пресликавање) и $s\circ\Pi^3_3 = \mathrm{Sup}^3_1(s;\Pi^3_3)$ (следбеник треће координате):

$$+ = \mathrm{Rec}^2(\Pi^1_1;\mathrm{Sup}^3_1(s;\Pi^3_3))\left|\begin{array}{l}+(x,0)=\Pi^1_1(x),\\ +(x,s(y))=s\circ\Pi^3_3(x,y,+(x,y));\end{array}\right.$$

Множење је добијено применом оператора Rec^2 на унарну нула-функцију $\mathbf{0}$ и функцију дужине три која је дефинисана као збир треће и прве координате, тј. као $\mathrm{Sup}^3_1(+;\Pi^3_3,\Pi^3_1)$. Дакле, $\cdot = \mathrm{Rec}^2(\mathbf{0},\mathrm{Sup}^3_1(+;\Pi^3_3,\Pi^3_1))$. Слично долазимо и до једнакости $\exp = \mathrm{Rec}^2(\mathbf{1},\mathrm{Sup}^3_2(\cdot;\Pi^3_3,\Pi^3_1))$.

[86] $f=\mathrm{Rec}^2(g,h)$:
$\left|\begin{array}{l}f(x,0)=g(x),\\ f(x,s(y))=h(x,y,f(x,y)),\end{array}\right.$

Користећи законе асоцијативности сабирања и множења, математичком индукцијом се може доказати да су за било које $k\geqslant 3$, RM-израчунљиве следеће k-арне функције:[87]

$$(x_1,\ldots,x_k)\mapsto x_1+\cdots+x_k=\sum_{i=1}^{k}x_i \text{ и } (x_1,\ldots,x_k)\mapsto x_1\cdot\ldots\cdot x_k=\prod_{i=1}^{k}x_i.$$

Због затворености за супституцију, ако су k-арне функције $g_1,\ldots,g_m$ RM-израчунљиве, такве су и функције:

$$(x_1,\ldots,x_k)\mapsto\sum_{i=1}^{m}g_i(\vec{x}) \text{ и } (x_1,\ldots,x_k)\mapsto\prod_{i=1}^{m}g_i(\vec{x}).$$

[87] На пример, за $k=3$, због асоцијативности, израз $x_1+x_2+x_3$ можемо сматрати краћим записом за $(x_1+x_2)+x_3$, па функцију $(x_1,x_2,x_3)\mapsto(x_1+x_2)+x_3$ описујемо на следећи начин: $\mathrm{Sup}^3_2(+;\mathrm{Sup}^3_2(+;\Pi^3_1,\Pi^3_2),\Pi^3_3)$. Доказ да су за свако $k\geqslant 3$ одговарајуће функције RM-израчунљиве спроводи се математичком индукцијом.

ПРИМЕР 27. *Ограничени збир и производ:* Ако је $f:\mathbb{N}^{k+1}\to\mathbb{N}$ било која RM-израчунљива функција, тада су функције $Z,P:\mathbb{N}^{k+1}\to\mathbb{N}$ дефинисане једнакостима:

$$\left|\begin{array}{l}Z(\vec{x},0)=0,\\ Z(\vec{x},y+1)\simeq Z(\vec{x},y)+f(\vec{x},y),\end{array}\right. \quad \text{и} \quad \left|\begin{array}{l}P(\vec{x},0)=1,\\ P(\vec{x},y+1)\simeq P(\vec{x},y)\cdot f(\vec{x},y).\end{array}\right.$$

такође RM-израчунљиве[88].

Уместо $Z(\vec{x},y)$ и $P(\vec{x},y)$ редом пишемо $\sum\limits_{z<y}f(\vec{x},z)$ и $\prod\limits_{z<y}f(\vec{x},z)$. Приметимо да је $\sum\limits_{z<0}f(\vec{x},z)=0$ и $\prod\limits_{z<0}f(\vec{x},z)=1$, за све $\vec{x}\in\mathbb{N}^k$.

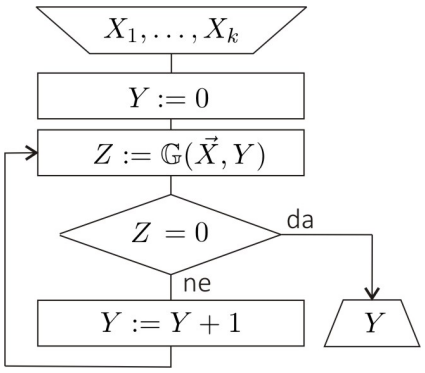
[88] $Z=\mathrm{Rec}(\mathbf{0}_k;h_z)$, при чему је h_z дато са $\mathrm{Sup}^{k+1}_2(+;\Pi^{k+2}_{k+2},\mathrm{Sup}^{k+1}_{k+1}(f;\Pi^{k+2}_1,\ldots,\Pi^{k+2}_{k+1}))$; $P=\mathrm{Rec}(\mathbf{1}_k;h_p)$, при чему је h_p дато са: $\mathrm{Sup}^{k+1}_2(\cdot;\Pi^{k+2}_{k+2},\mathrm{Sup}^{k+1}_{k+1}(f;\Pi^{k+2}_1,\ldots,\Pi^{k+2}_{k+1}))$.

Ако су $f,g:\mathbb{N}^{k+1}\to\mathbb{N}$ неке RM-израчунљиве, такве су и функције:

$$(\vec{x},y)\mapsto\sum_{z<g(\vec{x},y)}f(\vec{x},z) \text{ и } (\vec{x},y)\mapsto\prod_{z<g(\vec{x},y)}f(\vec{x},z)$$

Неограничена минимизација

Уопштено говорећи, неограничена минимизација одговара поступку потраге за решењем x једначине облика $g(x_1,\ldots,x_n,x)=0$, где су $x_1,\ldots,x_k$ параметри, а g је нека $(k+1)$-арна RM-израчунљива функција. Нека је $\mathbb{G}$ RM-програм који израчунава g. Саставимо програм $\mathbb{F}$ који, за задате $x_1,\ldots,x_n$, тражи најмање решење x једначине $g(x_1,\ldots,x_n,x)=0$. Прецизније, $\mathbb{F}$ за улаз $\vec{x}\in\mathbb{N}^k$ спроводи следећи поступак:



1. рачуна $g(\vec{x},0)$, користећи програм $\mathbb{G}$; уколико $\mathbb{G}(\vec{x},0)\downarrow$ и $g(\vec{x},0)\neq 0$, онда прелази на наредни корак, а ако $\mathbb{G}(\vec{x},0)\downarrow$ и $g(\vec{x},0)=0$, зауставља се и враћа излаз 0;

2. рачуна $g(\vec{x},1)$; ако $\mathbb{G}(\vec{x},1)\downarrow$ и $g(\vec{x},1)\neq 0$, онда прелази на наредни корак, а ако $\mathbb{G}(\vec{x},1)\downarrow$ и $g(\vec{x},1)=0$, зауставља се и враћа излаз 1;

3. рачуна $g(\vec{x},2)$; ако $\mathbb{G}(\vec{x},2)\downarrow$ и $g(\vec{x},2)\neq 0$, онда прелази на наредни корак, а ако $\mathbb{G}(\vec{x},2)\downarrow$ и $g(\vec{x},2)=0$, зауставља се и враћа излаз 2;

$\vdots$

Приметимо да је могуће да се наведени поступак никада не заврши и за то постоје два разлога:

- „заглављивање“ у неком кораку због незаустављања програма $\mathbb{G}$;
- успешно се рачунају све вредности функције g (редом, $g(\vec{x},0)$, $g(\vec{x},1)$, $g(\vec{x},2)$, ...), али никада се не добија вредност 0 (јер је $g(\vec{x},0)\neq 0, g(\vec{x},1)\neq 0, g(\vec{x},2)\neq 0,\ldots$).

Другим речима,

$$\mathbb{F}(\vec{x})\downarrow y \text{ акко } \mathbb{G}(\vec{x},x)\downarrow \text{ за свако } x\leqslant y,\quad g(\vec{x},x)\neq 0 \text{ за свако } x<y \text{ и } g(\vec{x},y)=0.$$

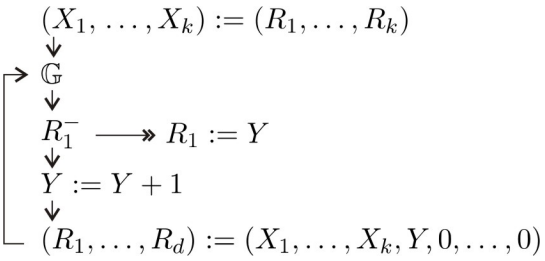
Неограничену минимизацију уводимо имајући на уму управо описани поступак. Ако је g нека $(k+1)$-арна парцијална функција, онда за k-арну функцију f такву да је

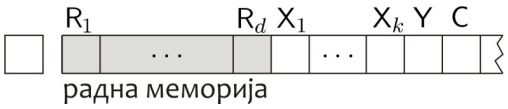
$$f(\vec{x})=\mu y(g(\vec{x},y)=0)=\begin{cases} y, & y \text{ је најмањи природан број} \\ & \text{такав да је } g(\vec{x},y)=0 \text{ и важи} \\ & (\forall z<y)(g(\vec{x},z)\downarrow \wedge g(\vec{x},z)\neq 0) \\ \uparrow, & \text{иначе.}\end{cases}$$

кажемо да је добијена (неограниченом) **минимизацијом** функције g, и пишемо $f=\mathrm{Min}^k(g)$.

Теорема 5. *Ако је RM-израчунљива $(k+1)$-арна функција g, онда је таква и функција* $\mathrm{Min}^k(g)$.

Доказ. Ако је $\mathbb{G}$ RM-програм који израчунава $(k+1)$-арну функцију g, нека је $d = \max\{k+1, ||\mathbb{G}||\}$. Низ регистара R_1, $\ldots$, R_d називаћемо *радном меморијом*. Регистре $R_{d+1}, \ldots, R_{d+k}$, R_{d+k+1}, R_{d+k+2} редом ћемо краће означити $X_1, \ldots, X_k, Y, C$, при чему је C регистар који користимо за копирање. Програм $\mathbb{F}$ за k-арну функцију $\mathrm{Min}^k(g)$ приказан је испод.





□

Пример 28. Да је функција

$$f(x) = \begin{cases} 0, & x = 0, \\ \uparrow, & x > 0, \end{cases}$$

RM-израчунљива следи из чињенице да се може описати на следећи начин: $f(x) = \mu y(x + y = 0)$, тј. $f = \mathrm{Min}^1(+)$.

Пример 29. Једноставном модификацијом доказа претходне теореме можемо доказати да је за било коју $(k+1)$-арну RM-израчунљиву функцију g и било коју константу $c \in \mathbb{N}$, функција

$$f_c(\vec{x}) = \mu y(g(\vec{x}, y) = c) = \begin{cases} y, & y \text{ је најмањи природан број} \\ & \text{такав да је } g(\vec{x}, y) = c \text{ и важи} \\ & (\forall z < y)(g(\vec{x}, z) \downarrow \wedge\, g(\vec{x}, z) \neq c) \\ \uparrow, & \text{иначе,} \end{cases}$$

такође RM-израчунљива.

Исто можемо показати и ослањајући се на једноставно проверљиву чињеницу да је RM-израчунљива функција $\chi_{\neq} : \mathbb{N}^2 \to \mathbb{N}$,

$$\chi_{\neq}(x, y) = \begin{cases} 1, & x \neq y, \\ 0, & x = y, \end{cases}$$

јер је тада RM-израчунљива и $(k+1)$-арна функција $(\vec{x}, y) \mapsto \chi_{\neq}(g(\vec{x}, y), c)$. Самим тим, RM-израчунљива је и функција f_c, јер важи

$$\mu y(\chi_{\neq}(g(\vec{x}, y), c) = 0) \simeq \mu y(g(\vec{x}, y) = c).$$

Пример 30. Функција $x \mapsto (\mu y(x + y = 3))^2$ је RM-израчунљива. Њен домен је коначан скуп $\{0, 1, 2, 3\}$:

$$(\mu y(0 + y = 3))^2 = 9,\ (\mu y(1 + y = 3))^2 = 4,\ (\mu y(2 + y = 3))^2 = 1,\ (\mu y(3 + y = 3))^2 = 0,$$

док за $x > 3$ није дефинисана: ако је $x > 3$, онда $(\mu y(x + y = 3))^2 \uparrow$.

Комбинатори и дрво израчунавања

Ако $\mathcal{F}_k$, $k \geqslant 1$, означава скуп свих k-арних парцијалних функција, онда супституцију, примитивну рекурзију и неограничену минимизацију одређују следећи **оператори**:

- $\mathrm{Sup}_m^k : \mathcal{F}_m \times \underbrace{\mathcal{F}_k \times \cdots \times \mathcal{F}_k}_{m \text{ пута}} \to \mathcal{F}_k$, $m \geqslant 1$;
- $\mathrm{Rec}^{k+1} : \mathcal{F}_k \times \mathcal{F}_{k+2} \to \mathcal{F}_{k+1}$;
- $\mathrm{Min}^k : \mathcal{F}_{k+1} \to \mathcal{F}_k$.

Полазећи од **основних функција**[90]:

- нула-функције $\mathbf{0} : \mathbb{N} \to \mathbb{N}$, $\mathbf{0}(x) = 0$,
- следбеника $s : \mathbb{N} \to \mathbb{N}$, $s(x) = x + 1$,
- пројекција $\Pi_i^k : \mathbb{N}^k \to \mathbb{N}$, $\Pi_i^k(x_1, \ldots, x_k) = x_i$ $(1 \leqslant i \leqslant k)$,

применом наведених оператора градимо веома богату класу парцијалних функција која је у центру пажње овог поглавља.

[90] Најједноставнији изрази језика аритметике одређују основне функције.

Дефиниција 6. ***Скуп парцијално рекурзивних функција*** *је најмањи скуп у смислу инклузије који садржи основне функције (нула-функцију, функцију следбеника и пројекције) и затворен је за супституцију, примитивну рекурзију и неограничену минимизацију.*

Све функције из претходног одељка чију смо RM-израчунљивост доказали применом леме 1 и теорема 3, 4 и 5 (видети примере 24, 26, 28) јесу парцијално рекурзивне.

Теорема 6. [91] *Свака парцијално рекурзивна функција је RM-израчунљива.*

[91] У наредном одељку показаћемо да важи и обрат теореме: свака RM-израчунљива функција је парцијално рекурзивна.

Парцијално рекурзивне функције описујемо помоћу израза које називамо **комбинаторима** и које формирамо (у складу са Дефиницијом 6) користећи ознаке за основне функције и ознаке за операторе супституције, примитивне рекурзије и неограничене минимизације (водећи посебно рачуна о дужини функција на које се примењују оператори). Комбинаторе прецизно уводимо на следећи начин, прецизирајући дужину (арност) сваког од њих:

- $\mathbf{0}$, s, Π_1^1 су унарни комбинатори, док су за $k > 1$, Π_i^k, $1 \leqslant i \leqslant k$ k-арни комбинатори;
- ако су $g_1, \ldots, g_m$ неки k-арни комбинатори и h неки m-арни комбинатор, онда је $\mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$ k-арни комбинатор;
- ако је g неки k-арни комбинатор и h неки $k+2$-арни комбинатор, онда је $\mathrm{Rec}^{k+1}(g, h)$ један $(k+1)$-арни комбинатор;

- ако је g неки $(k+1)$-арни комбинатор, онда је $\mathrm{Min}^k(g)$ један k-арни комбинатор.

Јасно, свака k-арна парцијално рекурзивна функција може се описати неким комбинатором и обрнуто, сваки комбинатор одређује једну парцијално рекурзивну функцију одговарајуће дужине. Изостављајући разне формалне детаље који се односе на комбинаторе, истичемо само најважнију чињеницу: сваки *комбинатор заправо описује поступак израчунавања вредности функције коју дефинише.*[92] Идентификујући комбинатор са функцијом коју дефинише, потпуно је природан опис поступка израчунавања. Ако је f неки k-арни комбинатор и $\vec{x} \in \mathbb{N}^k$, онда успешно израчунавање вредности $f(\vec{x})$ потврђује једнакост $f(\vec{x}) = z$, за неко $z \in \mathbb{N}$:

[92] Трагање за комбинатором неке задате функције може се схватити као настојање да се опише поступак израчунавања вредности те функције. Другим речима, комбинаторе можемо посматрати и као специфичне програме за израчунавање вредности функција.

- која се непосредно добија у случају основних комбинатора:

$$\mathbf{0}(x) = 0,\ s(x) = x+1,\ \Pi_i^k(x_1, \ldots, x_k) = x_i,$$

- а у случају сложенијих комбинатора последица је једнакости у којима се појављују једноставнији комбинатори (комбинатори мање сложености):
  - ако $f = \mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$, онда је једнакост $f(\vec{x}) = z$ последица једнакости

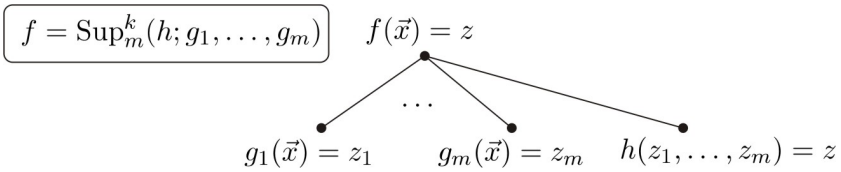
$$g_1(\vec{x}) = z_1, \ldots, g_m(\vec{x}) = z_m, h(z_1, \ldots, z_m) = z;$$

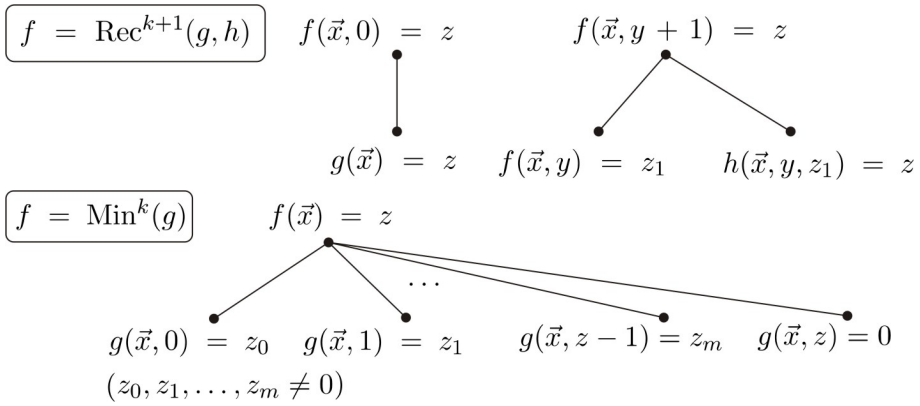
  - ако $f = \mathrm{Rec}^{k+1}(g,h)$, онда је једнакост $f(\vec{x},0) = z$ последица једнакости $g(\vec{x}) = z$, док је $f(\vec{x},y+1) = z$ последица једнакости $f(\vec{x},y) = z_1$ и $h(\vec{x},y,z_1) = z$;
  - ако $f = \mathrm{Min}^k(g)$, онда је једнакост $f(\vec{x}) = z$ последица једнакости $g(\vec{x},0) = z_0$, $g(\vec{x},1) = z_1$, ..., $g(\vec{x},z \dot{-} 1) = z_m$, $g(\vec{x},z) = 0$, за неке $z_0, z_1, \ldots, z_m$ различите од нуле.

Успешно израчунавање вредности $f(\vec{x})$ може се приказати и у облику тзв. **дрвета израчунавања** чији су чворови означени одговарајућим једнакостима, при чему је *корен дрвета* једнакост $f(\vec{x}) = z$, за неко $z \in \mathbb{N}$, *листови дрвета* (чворови без наследника) су облика:

$$\mathbf{0}(x) \overset{\bullet}{=} 0 \qquad s(x) \overset{\bullet}{=} x+1 \qquad \Pi_i^k(x_1, \ldots, \overset{\bullet}{x_k}) = x_i$$

а *унутрашњи чворови дрвета* (чворови који имају наследнике) су следећих облика:



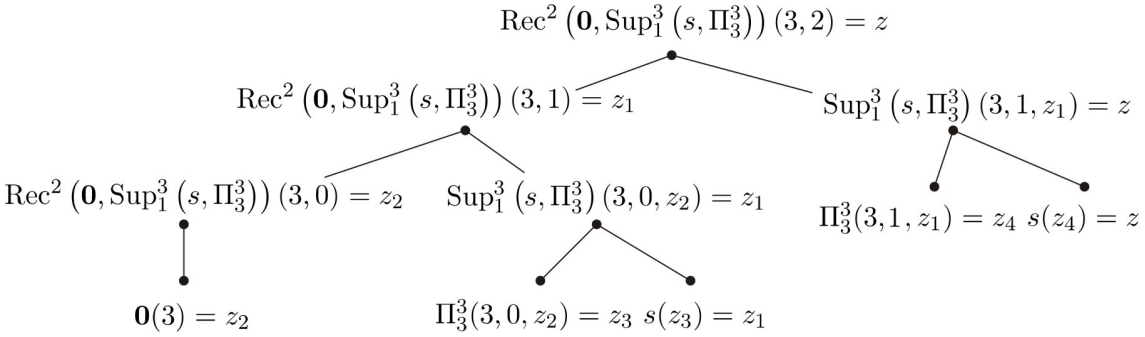


Наравно, $f(\vec{x}) \downarrow$ акко постоји дрво израчунавања вредности $f(\vec{x})$.

Пример 31. Да бисмо одредили дрво израчунавања комбинатора

$$\mathrm{Rec}^2\left(\mathbf{0}, \mathrm{Sup}_1^3\left(s, \Pi_3^3\right)\right)$$

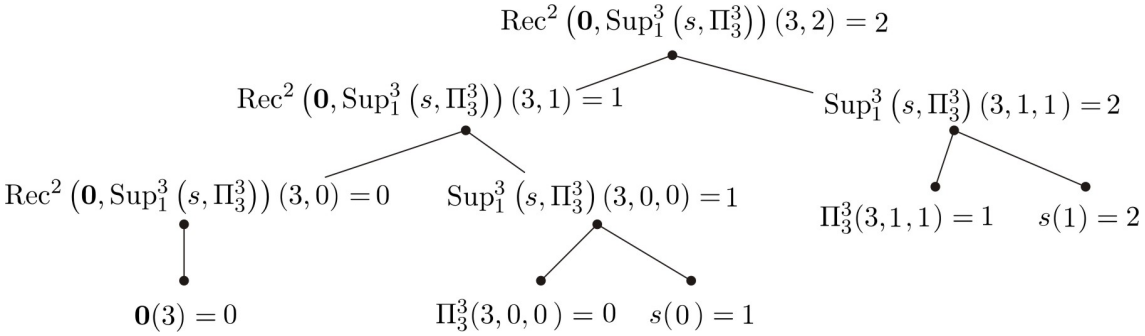
за аргументе (улаз) $(3,2)$, корисно је најпре формирати „скелет“ дрвета.



На основу скелета, једноставно је израчунати да је

$$\mathrm{Rec}^2\left(\mathbf{0}, \mathrm{Sup}_1^3\left(s, \Pi_3^3\right)\right)(3,2) = 2.$$

Одговарајуће дрво израчунавања је приказано на слици испод.



Примитивно рекурзивне функције и релације

Полазећи од основних функција, применом оператора супституције и примитивне рекурзије градимо довољно богат скуп функција за тзв. *аритметизацију* (тј. превођење на језик аритметике) бројних индуктивних дефиниција које се користе у математици.

Касније ћемо показати да се свака парцијално рекурзивна функција може дефинисати применом супституције и примитивне рекурзије уз највише једну примену неограничене минимизације.

Дефиниција 7. ***Скуп примитивно рекурзивних функција*** *је најмањи скуп у смислу инклузије који садржи основне функције (нула-функцију, функцију следбеника и пројекције) и затворен је за супституцију и примитивну рекурзију.*

Издвајамо очигледне особине примитивно рекурзивних функција:

1. Свака примитивно рекурзивна функција је тотална, јер су основне функције тоталне и применом оператора супституције и примитивне рекурзије на тоталне функције добијамо тоталне функције.

2. Свака примитивно рекурзивна функција је парцијално рекурзивна, па је тиме RM-, одн. TM-израчунљива.

У претходним одељцима смо већ навели примере примитивно рекурзивних функција:[94]

[94] Тврдње о примитивној рекурзивности неких важних функција нећемо посебно издвајати (као теореме), али ћемо их у овом одељку означавати скраћеницом **pr**.

(**pr1**) За произвољне $k \geqslant 1$ и $m \in \mathbb{N}$, константна k-арна функција $\mathbf{m}_k : \mathbb{N}^k \to \mathbb{N}$, $\mathbf{m}_k(\vec{x}) = m$ јесте примитивно рекурзивна.

(**pr2**) Сабирање, множење и степеновање јесу примитивно рекурзивне функције. Такође, ако су $g_1, \ldots, g_m : \mathbb{N}^k \to \mathbb{N}$ примитивно рекурзивне функције, онда су примитивно рекурзивне и функције $\vec{x} \mapsto \sum_{i=1}^{m} g_i(\vec{x})$ и $\vec{x} \mapsto \prod_{i=1}^{m} g_i(\vec{x})$.

(**pr3**) Ако су $f, g : \mathbb{N}^{k+1} \to \mathbb{N}$ примитивно рекурзивне функције, онда су такве и $(\vec{x}, y) \mapsto \sum\limits_{z<g(\vec{x},y)} f(\vec{x}, z)$ и $(\vec{x}, y) \mapsto \prod\limits_{z<g(\vec{x},y)} f(\vec{x}, z)$.

Пре него што наведемо нове примере примитивно рекурзивних функција, доказаћемо да су скупови примитивно рекурзивних и парцијално рекурзивних функција затворени за оператор Rec^1, који омогућава дефинисање унарних функција примитивном рекурзијом.

Теорема 7. *(1) Ако је $g \in \mathbb{N}$ произвољна константа и $h : \mathbb{N}^2 \to \mathbb{N}$ примитивно рекурзивна функција, онда је и функција $f : \mathbb{N} \to \mathbb{N}$ дата са:*

$$(\star) \left| \begin{array}{l} f(0) = g, \\ f(s(x)) = h(x, f(x)), \end{array} \right.$$

такође примитивно рекурзивна.

(2) Ако је $g \in \mathbb{N}$ произвољна константа и h бинарна парцијално рекурзивна функција, онда је и унарна функција f дата са:

$$\left| \begin{array}{l} f(0) = g, \\ f(s(x)) \simeq h(x, f(x)), \end{array} \right.$$

такође парцијално рекурзивна.

У оба случаја, претходне теореме, кажемо да је функција f добијена **примитивном рекурзијом константе g и бинарне функције h** и пишемо $f = \mathrm{Rec}^1(g, h)$.

Доказ. (1) Нека је $G : \mathbb{N} \to \mathbb{N}$ унарна константна функција, $G(x_1) = g$, а $H : \mathbb{N}^3 \to \mathbb{N}$ функција дата са $H(x_1, x_2, x_3) = h(x_2, x_3)$. Обе функције су примитивно рекурзивне, па је таква и функција $F = \mathrm{Rec}^2(G, H)$:

$$\begin{array}{l} F(x_1, 0) = G(x_1) = g, \\ F(x_1, s(x_2)) = H(x_1, x_2, F(x_1, x_2)) = h(x_2, F(x_1, x_2)). \end{array}$$

Није тешко доказати да је $f(x) = F(0, x)$, $x \in \mathbb{N}$. Ова једнакост следи из чињенице да услови $(\star)$ одређују јединствену функцију, а да функција $x \mapsto F(0, x)$ задовољава те услове:

$$\begin{array}{l} F(0, 0) = G(0) = g, \\ F(0, s(x)) = H(0, x, F(0, x)) = h(x, F(0, x)). \end{array}$$

Дакле, f је примитивно рекурзивна.

(2) Доказ је потпуно аналоган доказу под (1). □

Применом претходне теореме, проширујемо списак примитивно рекурзивних функција.

(**pr4**) **Факторијел** је примитиво рекурзивна функција:[95]

[95] $! = \mathrm{Rec}^1(1, \mathrm{Sup}_2^2(\cdot; \Pi_2^2, \mathrm{Sup}_1^2(s; \Pi_1^2)))$

$$\left| \begin{array}{l} 0! = 1, \\ (s(x))! = x! \cdot s(x). \end{array} \right.$$

(**pr5**) **Претходник** је примитивно рекурзивна функција:[96]

[96] $\mathrm{pd} = \mathrm{Rec}^1(0, \Pi_1^2)$

$$\mathrm{pd} : \mathbb{N} \to \mathbb{N} \left| \begin{array}{l} \mathrm{pd}(0) = 0, \\ \mathrm{pd}(s(x)) = x. \end{array} \right.$$

Уместо $\mathrm{pd}(x)$ писаћемо $x \dot{-} 1$.

(**pr6**) **Монус** је примитивно рекурзивна функција:[97]

[97] $\dot{-} = \mathrm{Rec}^2(\Pi_1^1, \mathrm{Sup}_1^3(\mathrm{pd}; \Pi_3^3))$

$$\dot{-} : \mathbb{N}^2 \to \mathbb{N} \left| \begin{array}{l} x \dot{-} 0 = x, \\ x \dot{-} s(y) = \mathrm{pd}(x \dot{-} y). \end{array} \right.$$

Дефиниција 8. *Скуп $R \subseteq \mathbb{N}^k$ (k-арна релација скупа $\mathbb{N}$) је примитивно рекурзиван(-на) ако је таква одговарајућа карактеристична функција $\chi_R : \mathbb{N}^k \to \mathbb{N}$,*

$$\chi_R(\vec{x}) = \begin{cases} 1, & \vec{x} \in R, \\ 0, & \vec{x} \notin R. \end{cases}$$

(**pr7**) Скупови $\mathbb{N}^+$ и $\{0\}$ јесу примитивно рекурзивни. За њихове карактеристичне функције $\chi_{\mathbb{N}^+}$ и $\chi_{\{0\}}$ уводимо посебне ознаке и називе: $\chi_{\mathbb{N}^+}$ називамо **функција знака** и означавамо $\mathrm{sg} : \mathbb{N} \to \mathbb{N}$, а $\chi_{\{0\}}$ називамо **функција обрнутог знака** и означавамо $\overline{\mathrm{sg}} : \mathbb{N} \to \mathbb{N}$. Једноставно се показује да су обе функције примитивно рекурзивне: [98]

$$\left| \begin{array}{l} \mathrm{sg}(0) = 0, \\ \mathrm{sg}(s(x)) = 1; \end{array} \right| \quad \left| \begin{array}{l} \overline{\mathrm{sg}}(0) = 1, \\ \overline{\mathrm{sg}}(s(x)) = 0. \end{array} \right.$$

[98] $\mathrm{sg} = \mathrm{Rec}^1(0, \mathbf{1}_2)$
$\overline{\mathrm{sg}} = \mathrm{Rec}^1(1, \mathbf{0}_2)$

(**pr8**) **Уређења** $\leqslant$ и $\geqslant$ су примитивно рекурзивне релације:[99]
$\chi_{\leqslant}(x,y) = \overline{\mathrm{sg}}(x \dot{-} y)$ и $\chi_{\geqslant}(x,y) = \overline{\mathrm{sg}}(y \dot{-} x)$.

Строга уређења $<$ и $>$ су такође примитивно рекурзивне релације:
$\chi_{<}(x,y) = 1 \dot{-} \chi_{\geqslant}(x,y)$ и $\chi_{>}(x,y) = 1 \dot{-} \chi_{\leqslant}(x,y)$.

[99] $\leqslant$ и $\geqslant$ посматрамо као подскупове од $\mathbb{N}^2$. Уместо $\chi_{\leqslant}$ пишемо leq, а уместо $\chi_{\geqslant}$ пишемо geq.

(**pr10**) **Једнакост** и **различитост** су примитивно рекурзивне релације:[100] $\chi_{=}(x,y) = \chi_{\leqslant}(x,y) \cdot \chi_{\geqslant}(x,y)$ и $\chi_{\neq}(x,y) = 1 \dot{-} \chi_{=}(x,y)$.

[100] Уместо $\chi_{=}$ пишемо eq.

Лема 2. *(1) **Комплемент** примитивно рекурзивног скупа је примитивно рекурзиван скуп: ако је $R \subseteq \mathbb{N}^k$ примитивно рекурзиван скуп, онда је примитивно рекурзиван и скуп $R^{\complement} = \mathbb{N}^k \setminus R$.*

*(2) **Унија и пресек** примитивно рекурзивних скупова (исте дужине) јесу примитивно рекурзивни скупови: ако су $P, Q \subseteq \mathbb{N}^k$ примитивно рекурзивни скупови, онда су примитивно рекурзивни и скупови $P \cap Q$ и $P \cup Q$.*

*(3) [**Ограничена квантификација.**] Ако је $R \subseteq \mathbb{N}^{k+1}$ примитивно рекурзивна релација, онда су примитивно рекурзивне и релације $U, E \subseteq \mathbb{N}^{k+1}$ дате са:*

$$U(\vec{x},y) \overset{\text{def}}{\Leftrightarrow} (\forall z \leqslant y) R(\vec{x},z) \;\; \textit{и} \;\; E(\vec{x},y) \overset{\text{def}}{\Leftrightarrow} (\exists z \leqslant y) R(\vec{x},z).$$

$$(\forall z \leqslant y) R(\vec{x},z) \Leftrightarrow R(\vec{x},0) \wedge R(\vec{x},1) \wedge \cdots \wedge R(\vec{x},y);$$
$$(\exists z \leqslant y) R(\vec{x},z) \Leftrightarrow R(\vec{x},0) \vee R(\vec{x},1) \vee \cdots \vee R(\vec{x},y)$$

Доказ. (1) $\chi_{R^{\complement}}(\vec{x}) = 1 \dot{-} \chi_R(\vec{x})$
(2) $\chi_{P \cap Q}(\vec{x}) = \chi_P(\vec{x}) \cdot \chi_Q(\vec{x})$; $P \cup Q = (P^{\complement} \cap Q^{\complement})^{\complement}$
(3) $\chi_U(\vec{x},y) = \prod\limits_{z \leqslant y} \chi_R(\vec{x},z)$; $\chi_E(\vec{x},y) = \mathrm{sg}\left(\sum\limits_{z \leqslant y} \chi_R(\vec{x},z) \right)$ □

Лема 3. *Ако су $f_1, \ldots, f_m : \mathbb{N}^k \to \mathbb{N}$ примитивно рекурзивне функције, и $R_1, \ldots, R_m \subseteq \mathbb{N}^k$ примитивно рекурзивне релације,*

такве да је $R_i \cap R_j = \emptyset$, $i \neq j$ *и* $\cup_{i=1}^{m} R_i = \mathbb{N}^k$, *онда је функција* $f : \mathbb{N}^k \to \mathbb{N}$,

$$f(\vec{x}) = \begin{cases} f_1(\vec{x}), & \text{ако је } R_1(\vec{x}), \\ \quad\vdots & \\ f_m(\vec{x}), & \text{ако је } R_m(\vec{x}), \end{cases}$$

примитивно рекурзивна.

Доказ. $f(\vec{x}) = \sum_{i=1}^{m} (f_i(\vec{x}) \cdot \chi_{R_i}(\vec{x}))$ □

Лема 4. [***Ограничена минимизација.***] *Ако је* $f : \mathbb{N}^{k+1} \to \mathbb{N}$ *примитивно рекурзивна функција, онда је и функција* $M : \mathbb{N}^{k+1} \to \mathbb{N}$,

$$M(\vec{x}, y) = (\mu z < y)(f(\vec{x}, z) = 0) = \begin{cases} u, & u < y, f(\vec{x}, u) = 0 \text{ и} \\ & (\forall v < u)\, f(\vec{x}, v) \neq 0, \\ y, & \text{иначе}, \end{cases}$$

примитивно рекурзивна.

На пример,

$$\begin{aligned} M(\vec{x}, 3) = {} & \mathrm{sg}(f(\vec{x}, 0)) + \\ & + \mathrm{sg}(f(\vec{x}, 0))\mathrm{sg}(f(\vec{x}, 1)) + \\ & + \mathrm{sg}(f(\vec{x}, 0))\mathrm{sg}(f(\vec{x}, 1))\mathrm{sg}(f(\vec{x}, 2)) \end{aligned}$$

Доказ. $M(\vec{x}, y) = \sum_{v<y} \prod_{u \leqslant v} \mathrm{sg}(f(\vec{x}, u))$ □

Последица 2. *Ако су* $f, g : \mathbb{N}^{k+1} \to \mathbb{N}$ *примитивно рекурзивне функције, онда је и* $(\vec{x}, y) \mapsto (\mu z < g(\vec{x}, y))(f(\vec{x}, z) = 0)$ *примитивно рекурзивна функција.*

Последица 3. *Ако је* $R \subseteq \mathbb{N}^{k+1}$ *примитивно рекурзивна релација, онда је и функција*

$$(\vec{x}, y) \mapsto (\mu z < g(\vec{x}, y))R(\vec{x}, z) \stackrel{\text{def}}{=} (\mu z < g(\vec{x}, y))(\chi_{R^{\complement}}(\vec{x}, z) = 0)$$

примитивно рекурзивна.

(**pr10**) **Количник**, уз договор да је $\left[\frac{y}{0}\right] = 0$, јесте примитивно рекурзивна функција:

$$\left[\frac{y}{x}\right] = \mathrm{sg}(x) \cdot (\mu k \leqslant y)(y < (k+1)x).$$

(**pr11**) **Остатак**, уз договор да је $\mathrm{rm}(0, y) = y$, јесте примитивно рекурзивна функција: $\mathrm{rm}(x, y) = y \dot{-} \left(\left[\frac{y}{x}\right] \cdot x\right)$.

(**pr12**) **Дељивост** је примитивно рекурзивна релација, јер важи $\chi_{|}(x, y) = \overline{\mathrm{sg}}(\mathrm{rm}(x, y))$. Уместо $\chi_{|}$ пишемо div.

(**pr13**) **Скуп простих бројева** $\mathbb{P} = \{2, 3, 5, 7, 11, \ldots\}$ је примитивно рекурзиван[103]:

$$\mathbb{P}(x) \Leftrightarrow x \geqslant 2 \wedge (\forall y \leqslant x)(y \mid x \Rightarrow y = 1 \vee y = x).$$

[103] $\chi_{\mathbb{P}}(x) = \mathrm{eq}\left(\sum_{k \leqslant x} \mathrm{div}(k, x), 2\right)$

(**pr14**) **Низ простих бројева**[104] је примитивно рекурзиван, јер се функција $p : \mathbb{N} \to \mathbb{N}$, неформално дефинисана са

$$p(x) = p_x = \text{„}(x+1)\text{-и прост број“},$$

може описати на следећи начин:

$$\begin{aligned} &p_0 = 2, \\ &p_{x+1} = (\mu y \leqslant p_x! + 1)(\mathbb{P}(y) \wedge y > p_x). \end{aligned}$$

[104] $p_0 = 2$, $p_1 = 3$, $p_2 = 5$, $p_3 = 7$, $p_4 = 11$, $p_5 = 13$, $p_6 = 17$, $p_7 = 19$, ...

(**pr15**) **Растављање на просте чиниоце** (видети страну 9): Функција

$$(x, y) \mapsto (y)_x = (\mu k \leqslant y)(p_x^k \mid y \wedge \neg(p_x^{k+1} \mid y))$$

је примитивно рекурзивна. Приметимо да је[105] $(y)_x$ изложилац простог броја p_x у канонској репрезентацији броја y.

[105] На пример, $1960 = 2^3 \cdot 5 \cdot 7^2$, па је $(1960)_0 = 3$, $(1960)_1 = 0$, $(1960)_2 = 1$, $(1960)_3 = 2$, $(1960)_x = 0$ за $x > 3$.

(**pr16**) Функција коју смо користили за **кодирање парова** природних бројева (страна 9), $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \overset{1-1}{\underset{\text{на}}{\to}} \mathbb{N}$,

$$\langle x_1, x_2 \rangle = 2^{x_1}(2x_2 + 1) - 1,$$

јесте примитивно рекурзивна. Инверзна функција $\langle \cdot, \cdot \rangle^{-1}$ одређена је примитивно рекурзивним функцијама $\langle \cdot \rangle_1, \langle \cdot \rangle_2 : \mathbb{N} \to \mathbb{N}$,

$$\langle x \rangle_1 = (x+1)_0, \ \langle x \rangle_2 = \left[\frac{\frac{x+1}{2^{(x+1)_0}} - 1}{2} \right], \ x \in \mathbb{N}.$$

Свако кодирање уређених парова природних бројева одређује по једно кодирање сваког од скупова $\mathbb{N}^k$, $k > 2$. На пример, кодирање скупа $\mathbb{N}^3$ одређује примитивно рекурзивна функција $(x_1, x_2, x_3) \mapsto \langle x_1, \langle x_2, x_3 \rangle \rangle$. Примитивно рекурзивне су и декодирајуће функције $x \mapsto \langle x \rangle_1$, $x \mapsto \langle \langle x \rangle_1 \rangle_2$, $x \mapsto \langle \langle x \rangle_2 \rangle_2$. Аналогно важи за свако фиксирано $k > 3$.

Теорема 8. [***Теорема симултане рекурзије***] *Ако су* $g_1, \ldots, g_m : \mathbb{N}^k \to \mathbb{N}$ *и* $h_1, \ldots, h_m : \mathbb{N}^{k+m+1} \to \mathbb{N}$ *примитивно рекурзивне функције, онда су једнакостима*

$$(\star) \begin{cases} f_1(\vec{x}, 0) = g_1(\vec{x}), \\ \vdots \\ f_m(\vec{x}, 0) = g_m(\vec{x}), \\ f_1(\vec{x}, y+1) = h_1(\vec{x}, y, f_1(\vec{x}, y), \ldots, f_m(\vec{x}, y)), \\ \vdots \\ f_m(\vec{x}, y+1) = h_m(\vec{x}, y, f_1(\vec{x}, y), \ldots, f_m(\vec{x}, y)), \end{cases}$$

на јединствен начин одређене функције $f_1, \ldots, f_m : \mathbb{N}^{k+1} \to \mathbb{N}$ *које су примитивно рекурзивне.*

Доказ. Наводимо доказ само у случају када је $m = 2$.

Да су функције f_1 и f_2 које задовољавају $(\star)$ јединствене доказује се математичком индукцијом.

Изаберимо неко кодирање скупа $\mathbb{N}^2$ одређено примитивно рекурзивним функцијама $\langle\cdot,\cdot\rangle : \mathbb{N}^2 \overset{1-1}{\underset{\text{на}}{\rightarrow}} \mathbb{N}$ и $\langle\cdot\rangle_i : \mathbb{N} \to \mathbb{N}$, $i = 1, 2$.[106]

Докажимо да постоје функције f_1 и f_2 које задовољавају $(\star)$.

Нека је $g(\vec{x}) = \langle g_1(\vec{x}), g_2(\vec{x})\rangle$ и

$$h(\vec{x}, y, z) = \langle h_1(\vec{x}, y, \langle z\rangle_1, \langle z\rangle_2), h_2(\vec{x}, y, \langle z\rangle_1, \langle z\rangle_2)\rangle.$$

Дефинишимо функцију $f : \mathbb{N}^{k+1} \to \mathbb{N}$ примитивном рекурзијом:

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}),\\ f(\vec{x}, y+1) &= h(\vec{x}, y, f(\vec{x}, y)).\end{aligned}$$

Није тешко доказати да функције $(\vec{x}, y) \mapsto \langle f(\vec{x}, y)\rangle_i$, $i = 1, 2$, задовољавају једнакости $(\star)$. Дакле, функције $f_i(\vec{x}, y) = \langle f(\vec{x}, y)\rangle_i$, $i = 1, 2$, јесу примитивно рекурзивне. □

[106] У општем случају бирамо кодирање скупа $\mathbb{N}^m$ одређено неким примитивно рекурзивним функцијама $\gamma : \mathbb{N}^m \overset{1-1}{\underset{\text{на}}{\rightarrow}} \mathbb{N}$ и $\gamma_i : \mathbb{N} \to \mathbb{N}$, $i = 1, \ldots, m$.

(**pr17**) **Итерација** је оператор који дефинишемо само за унарне функције. Ако је $f : \mathbb{N} \to \mathbb{N}$ примитивно рекурзивна функција, онда је и функција $F : \mathbb{N}^2 \to \mathbb{N}$ дата са

$$\begin{aligned} F(x, 0) &= x,\\ F(x, y+1) &= f(F(x, y)),\end{aligned}$$

примитивно рекурзивна. Кажемо да је функција F дефинисана *итерацијом* функције f.[107]

[107] Уместо $F(x, y)$ пише се $f^y(x)$: $f^0(x) = x$, $f^{y+1}(x) = f(f^y(x))$.

Теорема 9. *Скуп примитивно рекурзивних функција је најмањи скуп функција који садржи основне функције и (било које примитивно рекурзивне) функције $\langle\cdot,\cdot\rangle$, $\langle\cdot\rangle_1$ и $\langle\cdot\rangle_2$ (које омогућавају кодирање парова природних бројева) и затворен је за супституцију и итерацију (наравно, унарних функција).*

Доказ. Нека је $\mathcal{C}$ најмањи скуп функција који задовољава набројане услове. Како су функције $\langle\cdot,\cdot\rangle$, $\langle\cdot\rangle_1$ и $\langle\cdot\rangle_2$ примитивно рекурзивне, и како је скуп примитивно рекурзивних функција затворена за итерацију, све функције из $\mathcal{C}$ јесу примитивно рекурзивне.

Преостаје да покажемо да је $\mathcal{C}$ затворен за примитивну рекурзију. Да бисмо избегли заморне техничке детаље, а истакли суштину доказа, показаћемо само следеће: ако $g \in \mathbb{N}$ и $h : \mathbb{N} \to \mathbb{N}$ припада $\mathcal{C}$, онда функција $f : \mathbb{N} \to \mathbb{N}$ дефинисана са:

$$\left|\begin{aligned} f(0) &= g,\\ f(x+1) &= h(x, f(x)).\end{aligned}\right.$$

такође припада $\mathcal{C}$.

У општем случају користи се чињеница да за свако $k \geqslant 2$, скуп $\mathcal{C}$ садржи функције које омогућавају кодирање и декодирање скупа $\mathbb{N}^k$.

Функција $H : \mathbb{N} \to \mathbb{N}$, $H(x) = \langle \langle x \rangle_1 + 1, h(\langle x \rangle_1, \langle x \rangle_2) \rangle$, очигледно припада $\mathcal{C}$. Такође, $\mathcal{C}$ садржи и функцију $F : \mathbb{N}^2 \to \mathbb{N}$ дефинисану итерацијом функције H:

$$\begin{aligned} F(x,0) &= x, \\ F(x,n+1) &= H(F(x,n)). \end{aligned}$$

($\star$) Докажимо прво да за свако $x \in \mathbb{N}$, $\langle F(\langle 0, g \rangle, x) \rangle_1 = x$.
База индукције: $x = 0$. $\langle F(\langle 0, g \rangle, 0) \rangle_1 = \langle \langle 0, g \rangle \rangle_1 = 0$.
Индуктивна претпоставка. Нека је $\langle F(\langle 0, g \rangle, x) \rangle_1 = x$. Тада је

$$\begin{aligned} \langle F(\langle 0, g \rangle, x+1) \rangle_1 &= \langle H(F(\langle 0, g \rangle, x)) \rangle_1 \\ &= \langle \langle F(\langle 0, g \rangle, x) \rangle_1 + 1, h(\langle F(\langle 0, g \rangle, x) \rangle_1, \langle F(\langle 0, g \rangle, x) \rangle_2) \rangle \rangle_1 \\ &= \langle F(\langle 0, g \rangle, x) \rangle_1 + 1 = x + 1 \end{aligned}$$

Најзад, доказаћемо да је $\langle F(\langle 0, g \rangle, x) \rangle_2 = f(x)$, $x \in \mathbb{N}$, одакле следи да $f \in \mathcal{C}$.
База индукције: $x = 0$. $\langle F(\langle 0, g \rangle, 0) \rangle_2 = \langle \langle 0, g \rangle \rangle_2 = g$.
Индуктивна претпоставка. Нека је $\langle F(\langle 0, g \rangle, x) \rangle_2 = f(x)$. Тада је

$$\begin{aligned} \langle F(\langle 0, g \rangle, x+1) \rangle_2 &= \langle H(F(\langle 0, g \rangle, x)) \rangle_2 \\ &= \langle \langle F(\langle 0, g \rangle, x) \rangle_1 + 1, h(\langle F(\langle 0, g \rangle, x) \rangle_1, \langle F(\langle 0, g \rangle, x) \rangle_2) \rangle \rangle_2 \\ &= h(\langle F(\langle 0, g \rangle, x) \rangle_1, \langle F(\langle 0, g \rangle, x) \rangle_2) \\ &= h(x, f(x)) \quad [\text{према } (\star) \text{ и индуктивној претпоставци}]. \end{aligned}$$

□

(**pr18**) **Бинарна репрезентација броја**: Сваки $x \in \mathbb{N}$ има јединствен приказ облика $x = \sum_{i=0}^{+\infty} c_{i,x} 2^i$, $c_i \in \{0,1\}$. Двоструко индексирани низ $(c_{i,x})$ јесте заправо једна примитивно рекурзивна функција $\mathbf{c} : \mathbb{N} \times \mathbb{N} \to \{0,1\}$, $\mathbf{c}(i,x) = \mathrm{rm}(2, \left[\frac{x}{2^i}\right])$.[109]

[109] $\mathbf{c}(i,x) = 0$ за $i \geqslant x$, јер за свако m важи $m < 2^m$.

Ако је $x > 0$, онда постоји јединствени приказ облика

$(*)$ $\quad x = 2^{b_1} + 2^{b_2} + \cdots + 2^{b_k}$, $0 \leqslant b_1 < b_2 < \ldots < b_k, k \geqslant 1$.

Функција $\mathrm{lh} : \mathbb{N} \to \mathbb{N}$, којом се за $x > 0$ одређује k у горњем приказу $(*)$:

$$\mathrm{lh}(x) = \begin{cases} k \text{ у приказу } (*), & x > 0, \\ 0, & x = 0, \end{cases}$$

јесте примитивно рекурзивна: $\mathrm{lh}(x) = \sum_{i<x} \mathbf{c}(i,x)$. Таква је и функција $\mathbf{b} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, којом се одређују изложиоци b_i:[110]

[110] $x = \sum_{j<b_1} 0 \cdot 2^j + 1 \cdot 2^{b_1} + \sum_{b_1<j<b_2} 0 \cdot 2^j + 1 \cdot 2^{b_2} + \cdots$

$$\begin{aligned} \mathbf{b}(i,x) &= \begin{cases} b_i \text{ у приказу } (*), & x > 0 \text{ и } 1 \leqslant i \leqslant \mathrm{lh}(x), \\ 0, & \text{иначе}, \end{cases} \\ &= \begin{cases} (\mu k < x)\left(\sum_{j \leqslant k} \mathbf{c}(j,x) = i\right), & x > 0 \text{ и } 1 \leqslant i \leqslant \mathrm{lh}(x), \\ 0, & \text{иначе}. \end{cases} \end{aligned}$$

Кодирање коначних низова природних бројева уведено је на страни 10:

$$\begin{aligned}\lceil()\rceil &= 0 \quad (\text{Кôд празног низа је } 0)\\ \lceil x_1,\dots,x_k\rceil &= 2^{x_1}+2^{x_1+x_2+1}+\dots+2^{x_1+x_2+\dots+x_k+k-1}\\ &= (1\underbrace{0\dots0}_{x_k\text{ нула}}1\dots1\underbrace{0\dots0}_{x_1\text{ нула}})_2\end{aligned}$$

$\lceil()\rceil = 0$
$\lceil x_1,\dots,x_k\rceil = |x_1, \lceil x_2,\dots,x_k\rceil|$
Функција $|\cdot,\cdot| : \mathbb{N}^2 \to \mathbb{N}^+$ дата са $|x_1,x_2| = 2^{x_1}(2x_2+1)$ је бијекција. Њена инверзна функција је одређена функцијама $|\cdot|_1, |\cdot|_2 : \mathbb{N}^+ \to \mathbb{N}$.

Нема смисла расправљати о примитивној рекурзивности функције $\lceil\cdot\rceil$, будући да је њен домен $\bigcup_{k\geqslant 0}\mathbb{N}^k$. Али, постоје примитивно рекурзивне функције којима се реконструише низ чији је кôд задат. Функцијом $\mathrm{lh} : \mathbb{N} \to \mathbb{N}$ рачунамо дужину низа чији је кôд задат,

$$\mathrm{lh}(x) = \text{ дужина низа чији је кôд } x,$$

док нам следећа примитивно рекурзивна функција омогућава одређивање чланова низа чији је кôд задат:

Ако је $x > 0$, онда је:
$x = 2^{\mathbf{b}(1,x)} + 2^{\mathbf{b}(2,x)} + \dots + 2^{\mathbf{b}(\mathrm{lh}(x),x)}$
$= 2^{\mathbf{a}(1,x)} + 2^{\mathbf{a}(1,x)+\mathbf{a}(2,x)+1} + \dots$
$\dots + 2^{\mathbf{a}(1,x)+\dots+\mathbf{a}(\mathrm{lh}(x),x)+\mathrm{lh}(x)\dot{-}1}$

$$\begin{aligned}(i,x) \mapsto \lfloor x\rfloor_i &= \begin{cases} i\text{-ти члан низа чији је кôд } x, & x>0, 1\leqslant i\leqslant \mathrm{lh}(x),\\ 0, & \text{иначе.}\end{cases}\\ &= \begin{cases} \mathbf{b}(i,x)\dot{-}(\mathbf{b}(i\dot{-}1,x)+1), & x>0, 1\leqslant i\leqslant \mathrm{lh}(x),\\ 0, & \text{иначе.}\end{cases}\end{aligned}$$

Подсећамо, ако је $x > 0$, онда је $x = \lceil \lfloor x\rfloor_1,\dots,\lfloor x\rfloor_{\mathrm{lh}(x)}\rceil$.

Ако је $f : \mathbb{N}^{k+1} \to \mathbb{N}$ произвољна функција, онда функцију $\widehat{f} : \mathbb{N}^{k+1} \to \mathbb{N}$ дефинисану са

$$\widehat{f}(\vec{x},y) = \lceil f(\vec{x},y),\dots,f(\vec{x},0)\rceil$$

називамо **историја** функције f. Наредна теорема тврди да је скуп примитивно рекурзивних функција затворен за рекурзију у којој вредност $f(\vec{x},y)$ зависи од свих вредности $f(\vec{x},z)$, $z < y$.

Теорема 10. [***Принцип тоталне рекурзије***] *Ако су* $g : \mathbb{N}^k \to \mathbb{N}$ *и* $h : \mathbb{N}^{k+2} \to \mathbb{N}$ *примитивно рекурзивне функције, онда је примитивно рекурзивна и функција* $f : \mathbb{N}^{k+1} \to \mathbb{N}$ *дефинисана са:*

$$\begin{aligned}f(\vec{x},0) &= g(\vec{x}),\\ f(\vec{x},y+1) &= h(\vec{x},y,\widehat{f}(\vec{x},y))\end{aligned}$$

Доказ. Дефинишимо најпре функцију $\widehat{f} : \mathbb{N}^{k+1} \to \mathbb{N}$ примитивном рекурзијом на следећи начин:

$$\begin{aligned}\widehat{f}(\vec{x},0) &= 2^{g(\vec{x})},\\ \widehat{f}(\vec{x},y+1) &= |h(\vec{x},y,\widehat{f}(\vec{x},y)),\widehat{f}(\vec{x},y)|.\end{aligned}$$

Функција $\widehat{f}$ је примитивно рекурзивна, па је примитивно рекурзивна и функција $f : \mathbb{N}^k \to \mathbb{N}$ дефинисана са $f(\vec{x},y) = \lfloor \widehat{f}(\vec{x},y)\rfloor_1$. □

Свака RM-израчунљива функција је парцијално рекурзивна

У овом одељку доказујемо да је свака RM-израчунљива функција парцијално рекурзивна.

Кључну улогу при извршавању било ког RM-програма $\mathbb{P}$ има функција $\mathsf{NEXT}_{\mathbb{P}} : \mathrm{Conf} \to \mathrm{Conf}$ (видети страну 29), коју овом приликом незнатно модификујемо претпостављајући да се након заустављања програма $\mathbb{P}$ у бројач уписује 0:

$$\mathsf{NEXT}_{\mathbb{P}}(i; r_1, \ldots, r_k, \ldots, r_d, 0 \cdots) = \begin{cases} (\ell; r_1, \ldots, r_k + 1, \ldots, r_d, 0 \cdots), & i\text{-та инструкција у } \mathbb{P} \text{ је } R_k^+ \mid \ell, \\ (\ell_0; r_1, \ldots, r_k, \ldots, r_d, 0 \cdots), & i\text{-та инструкција у } \mathbb{P} \text{ је } R_k^- \mid \ell_0, \ell_1 \text{ и } r_k = 0, \\ (\ell_1; r_1, \ldots, r_k - 1, \ldots, r_d, 0 \cdots), & i\text{-та инструкција у } \mathbb{P} \text{ је } R_k^- \mid \ell_0, \ell_1 \text{ и } r_k \neq 0, \\ (0; r_1, \ldots, r_k, \ldots, r_d, 0 \cdots), & i\text{-та инструкција у } \mathbb{P} \text{ не постоји.} \end{cases}$$

Будући да скуп свих конфигурација Conf идентификујемо са скупом свих низова природних бројева чији су сви чланови почев од неког једнаки нули, на природан начин можемо кодирати све конфигурације из Conf природним бројевима:[113]

$$(i, r_1, r_2, \ldots, r_d, 0, 0, 0, \ldots) \mapsto p_0^i p_1^{r_1} \cdots p_d^{r_d}.$$

[113] Свакој конфигурацији из Conf једнозначно придружујемо природан број из $\mathbb{N}^+$ – кôд те конфигурације.

Ослањајући се на уведено кодирање конфигурација, аритметизоваћемо функцију $\mathrm{NEXT}_{\mathbb{P}}$, тј. дефинисаћемо примитивно рекурзивну функцију $\mathrm{next}_{\mathbb{P}} : \mathbb{N} \to \mathbb{N}$, такву да за сваку конфигурацију C важи: $\mathrm{next}_{\mathbb{P}}(\text{кôд}(C)) = \text{кôд}(\mathrm{NEXT}_{\mathbb{P}}(C))$. Другим речима, број $\mathrm{next}_{\mathbb{P}}(c)$ је кôд конфигурације добијене применом програма $\mathbb{P}$ на конфигурацију чији је кôд c.

$$\begin{array}{ccccc} & \mathrm{Conf} & \xrightarrow{\mathrm{NEXT}_{\mathbb{P}}} & \mathrm{Conf} & \\ \text{кôд} & \downarrow & & \downarrow & \text{кôд} \\ & \mathbb{N} & \xrightarrow{\mathrm{next}_{\mathbb{P}}} & \mathbb{N} & \end{array}$$

- Ако је у програму $\mathbb{P}$ i-та инструкција $R_k^+ \mid \ell$:

$$\left.\begin{array}{lcl} C = i, r_1, \ldots, r_k, \ldots & \mapsto & c = p_0^i p_1^{r_1} \cdots p_k^{r_k} \cdots \\ \downarrow \mathrm{NEXT}_{\mathbb{P}} & & \\ C' = \ell, r_1, \ldots, r_k + 1, \ldots & \mapsto & c' = p_0^{\ell} p_1^{r_1} \cdots p_k^{r_k+1} \cdots \end{array}\right\} \mathrm{next}_{\mathbb{P}}(c) = c' = \frac{c}{p_0^{(c)_0}} \cdot p_0^{\ell} \cdot p_k$$

- Ако је у програму $\mathbb{P}$ i-та инструкција $R_k^- \mid \ell_1, \ell_2$ и $r_k \neq 0$:

$$\left.\begin{array}{lcl} C = i, r_1, \ldots, r_k, \ldots & \mapsto & c = p_0^i p_1^{r_1} \cdots p_k^{r_k} \cdots \\ \downarrow \mathrm{NEXT}_{\mathbb{P}} & & \\ C' = \ell_2, r_1, \ldots, r_k - 1, \ldots & \mapsto & c' = p_0^{\ell_2} p_1^{r_1} \cdots p_k^{r_k-1} \cdots \end{array}\right\} \mathrm{next}_{\mathbb{P}}(c) = c' = \frac{c}{p_0^{(c)_0} \cdot p_k} \cdot p_0^{\ell_2}$$

- Ако је у програму $\mathbb{P}$ i-та инструкција $R_k^- \mid \ell_1, \ell_2$ и $r_k = 0$:

$$\left.\begin{array}{lcl} C = i, r_1, \ldots, r_k, \ldots & \mapsto & c = p_0^i p_1^{r_1} \cdots p_k^{r_k} \cdots \\ \downarrow \mathrm{NEXT}_{\mathbb{P}} & & \\ C' = \ell_1, r_1, \ldots, r_k, \ldots & \mapsto & c' = p_0^{\ell_1} p_1^{r_1} \cdots p_k^{r_k} \cdots \end{array}\right\} \mathrm{next}_{\mathbb{P}}(c) = c' = \frac{c}{p_0^{(c)_0}} \cdot p_0^{\ell_1}$$

- Ако у програму $\mathbb{P}$ не постоји i-та инструкција:

$$\left.\begin{array}{lcl} C = i, r_1, \ldots, r_k, \ldots & \mapsto & c = p_0^i p_1^{r_1} \cdots p_k^{r_k} \cdots \\ \downarrow \mathrm{NEXT}_{\mathbb{P}} & & \\ C' = 0, r_1, \ldots, r_k, \ldots & \mapsto & c' = p_0^0 p_1^{r_1} \cdots p_k^{r_k} \cdots \end{array}\right\} \mathrm{next}_{\mathbb{P}}(c) = c' = \frac{c}{p_0^{(c)_0}}$$

Све у свему, $\text{next}_{\mathbb{P}} : \mathbb{N} \to \mathbb{N}$ јесте функција дата са:

$$(\star) \qquad \text{next}_{\mathbb{P}}(c) = \begin{cases} \dfrac{c}{p_0^{(c)_0}} \cdot p_0^{\ell} \cdot p_k, & (c)_0\text{-та инструкција у } \mathbb{P} \text{ је } R_k^+ \mid \ell, \\ \dfrac{c}{p_0^{(c)_0}} \cdot p_0^{\ell_1}, & (c)_0\text{-та инструкција у } \mathbb{P} \text{ је } R_k^- \mid \ell_1, \ell_2 \text{ и } (c)_k = 0, \\ \dfrac{c}{p_0^{(c)_0} \cdot p_k} \cdot p_0^{\ell_2}, & (c)_0\text{-та инструкција у } \mathbb{P} \text{ је } R_k^- \mid \ell_1, \ell_2 \text{ и } (c)_k \neq 0, \\ \dfrac{c}{p_0^{(c)_0}}, & (c)_0\text{-та инструкција у } \mathbb{P} \text{ не постоји.} \end{cases}$$

Лема 5. *За сваки RM-програм* $\mathbb{P}$ *функција* $\text{next}_{\mathbb{P}} : \mathbb{N} \to \mathbb{N}$ *дефинисана са* $(\star)$ *јесте примитивно рекурзивна.*

ПРИМЕР 32. Ако је $\mathbb{P}$ програм $\begin{cases} 1. & R_2^- \mid 3, 2 \\ 2. & R_1^+ \mid 1 \end{cases}$ онда је

$$\text{next}_{\mathbb{P}}(c) = \begin{cases} \dfrac{c}{p_0^{(c)_0} \cdot p_2} \cdot p_0^2, & (c)_0 = 1 \wedge (c)_2 \neq 0, \\ \dfrac{c}{p_0^{(c)_0}} \cdot p_0^3, & (c)_0 = 1 \wedge (c)_2 = 0, \\ \dfrac{c}{p_0^{(c)_0}} \cdot p_0^1 \cdot p_1, & (c)_0 = 2, \\ \dfrac{c}{p_0^{(c)_0}}, & (c)_0 = 0 \vee (c)_0 > 2. \end{cases} = \begin{cases} 2 \cdot \dfrac{c}{5}, & (c)_0 = 1 \wedge (c)_2 \neq 0, \\ 2^2 \cdot c, & (c)_0 = 1 \wedge (c)_2 = 0, \\ 3 \cdot \dfrac{c}{2}, & (c)_0 = 2, \\ \dfrac{c}{2^{(c)_0}}, & (c)_0 = 0 \vee (c)_0 > 2. \end{cases}$$

Приметимо да извршавање програма $\mathbb{P}$ у потпуности описује узастопна примена функције $\text{next}_{\mathbb{P}}$ на кôд полазне конфигурације.

1	3	2	$0\cdots$	$2^1 \cdot 3^3 \cdot 5^2 (= 1350)$
2	3	1	$0\cdots$	$\text{next}_{\mathbb{P}}(2^1 \cdot 3^3 \cdot 5^2) = 2^2 \cdot 3^3 \cdot 5^1$
1	4	1	$0\cdots$	$\text{next}_{\mathbb{P}}(2^2 \cdot 3^3 \cdot 5^1) = 2^1 \cdot 3^4 \cdot 5^1$
2	4	0	$0\cdots$	$\text{next}_{\mathbb{P}}(2^1 \cdot 3^4 \cdot 5^1) = 2^2 \cdot 3^4 \cdot 5^0$
1	5	0	$0\cdots$	$\text{next}_{\mathbb{P}}(2^2 \cdot 3^4 \cdot 5^0) = 2^1 \cdot 3^5 \cdot 5^0$
3	5	0	$0\cdots$	$\text{next}_{\mathbb{P}}(2^1 \cdot 3^5 \cdot 5^0) = 2^3 \cdot 3^5 \cdot 5^0$
0	5	0	$0\cdots$	$\text{next}_{\mathbb{P}}(2^3 \cdot 3^5 \cdot 5^0) = 2^0 \cdot 3^5 \cdot 5^0$

Теорема 11. *Свака RM-израчунљива функција је парцијално рекурзивна.*

ДОКАЗ. Нека је f произвољна k-арна RM-израчунљива функција и $\mathbb{P}$ неки RM-програм који израчунава f. За задато $\vec{x} \in \mathbb{N}^k$, поступак израчунавања вредности $f(\vec{x})$, помоћу програма $\mathbb{P}$ јесте низ рачунских корака у којима се мењају конфигурације. Функцију $\text{Comp}_{\mathbb{P}} : \mathbb{N}^{k+1} \to \mathbb{N}$, неформално описану са

$$\text{Comp}_{\mathbb{P}}(\vec{x}, t) = \text{кôд конфигурације која се достиже после } t \text{ корака израчунавања програма } \mathbb{P} \text{ за улаз } \vec{x},$$

прецизно дефинишемо на следећи начин[115]:

[115] $\text{Comp}_{\mathbb{P}}(\vec{x}, 0)$ је кôд почетне конфигурације за улаз $\vec{x}$.

$$\begin{cases} \mathrm{Comp}_{\mathbb{P}}(\vec{x},0) = p_0^1 p_1^{x_1} \cdots p_k^{x_k}, \\ \mathrm{Comp}_{\mathbb{P}}(\vec{x},t+1) = \mathrm{next}_{\mathbb{P}}(\mathrm{Comp}_{\mathbb{P}}(\vec{x},t)). \end{cases}$$

Како је $\mathrm{next}_{\mathbb{P}}$ примитивно рекурзивна функција, и $\mathrm{Comp}_{\mathbb{P}}$ је примитивно рекурзивна. Дефинишимо и k-арну парцијално рекурзивну функцију $\mathrm{Halt}_{\mathbb{P}}$:

$$\mathrm{Halt}_{\mathbb{P}}(\vec{x}) = \mu t \left((\mathrm{Comp}_{\mathbb{P}}(\vec{x},t))_0 = 0 \right).$$

Јасно, ако $\mathrm{Halt}_{\mathbb{P}}(\vec{x}) \downarrow$, онда се израчунавање програма $\mathbb{P}$ за улаз $\vec{x}$ зауставља (за мање од $\mathrm{Halt}_{\mathbb{P}}(\vec{x})$ корака). Тада је садржај првог регистра у завршној конфигурацији резултат израчунавања програма $\mathbb{P}$ за улаз $\vec{x}$. Како $\mathbb{P}$ израчунава функцију f следи да је

$$f(\vec{x}) = (\mathrm{Comp}_{\mathbb{P}}(\vec{x}, \mathrm{Halt}_{\mathbb{P}}(\vec{x})))_1.$$

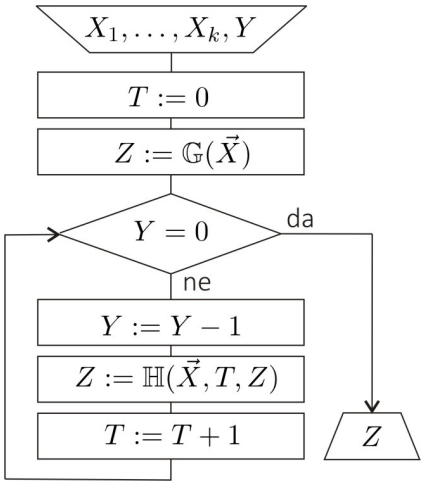
Дакле, f јесте парцијално рекурзивна. □

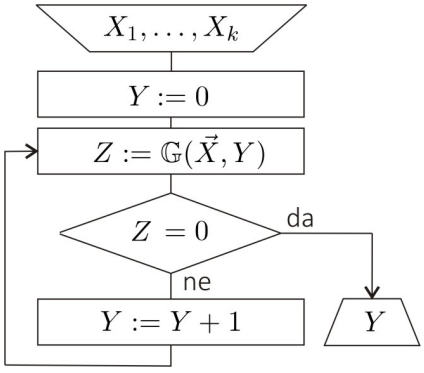
Последица 4. *Функција је* RM*-израчунљива акко је парцијално рекурзивна.*

Последица 5. *Свака парцијално рекурзивна функција може се дефинисати комбинатором у коме се појављује највише једна неограничена минимизација.*

WHILE-*програми*

RM-програми могли би се уврстити у тзв. *неструктуиране програме*, будући да их карактеришу goto-запетљавања. Ако пажљивије анализирамо програме који одговарају операторима супституције, примитивне рекурзије и неограничене минимизације (видети доказе Теорема 3, 4 и 5) уочићемо да, поред надовезивања програма, *циклуси* играју кључну улогу. Ово запажање доводи до идеје да израчунљивост парцијално рекурзивних функција можемо окарактерисати и *структуираним* програмима, какви су while-програми које уводимо индуктивно на следећи начин:





- $\mathsf{R}_k := \mathsf{R}_k + 1$ је while-програм дубине 0 (за било које $k \geqslant 1$);
- $\mathsf{R}_k := \mathsf{R}_k \dot{-} 1$ је while-програм дубине 0 (за било које $k \geqslant 1$);
- ако су $\mathbb{W}_1$ и $\mathbb{W}_2$ while-програми редом дубина n_1 и n_2, онда је и $(\mathbb{W}_1; \mathbb{W}_2)$ while-програм дубине $\max\{n_1, n_2\}$;
- ако је $\mathbb{W}$ while-програм дубине n, онда је и (while $\mathsf{R}_k \neq 0$ do $\mathbb{W}$) while-програм дубине $n+1$ (за било које $k \geqslant 1$).

Претпостављамо да је очигледно на који начин регистар машина извршава while-програме и истичемо само да за овакве програме

машини није потребан бројач (за редни број инструкције), јер инструкције while-програма нису нумерисане. Дакле, у овом случају конфигурације су одређене само садржајем регистара. На природан начин дефинишемо и појам while-израчунљиве функције. Парцијална k-арна функција f је while-**израчунљива** ако постоји while-програм $\mathbb{W}$ такав да за све $\vec{x} \in \mathbb{N}^k$ важи:

- ако $\vec{x} \in \mathrm{dom}(f)$, онда се извршавање програма $\mathbb{W}$ за улаз $\vec{x}$ зауставља и садржај првог регистра у завршној конфигурацији је $f(\vec{x})$;
- ако $\vec{x} \notin \mathrm{dom}(f)$, онда се извршавање програма $\mathbb{W}$ за улаз $\vec{x}$ не зауставља.

ПРИМЕР 33. Сабирање $+ : \mathbb{N}^2 \to \mathbb{N}$ је while-израчунљива функција:

$$(\mathsf{while}\ \mathsf{R}_2 \neq 0\ \mathsf{do}\ (\mathsf{R}_1 := \mathsf{R}_1 + 1;\ \mathsf{R}_2 := \mathsf{R}_2 \dot{-} 1))$$

Празна функција је while-израчунљива:

$$(\mathsf{R}_1 := \mathsf{R}_1 + 1;\ (\mathsf{while}\ \mathsf{R}_1 \neq 0\ \mathsf{do}\ \mathsf{R}_1 := \mathsf{R}_1 + 1))$$

ВЕЖБАЊЕ. Доказати да су све парцијално рекурзивне функције while-израчунљиве.

ВЕЖБАЊЕ. Доказати да се сваки while-програм може симулирати неким RM-програмом.[117]

[117] Упутство: доказати математичком индукцијом по дубини while-програма.

На основу тврђења наведених у претходним вежбањима закључујемо да је скуп свих while-израчунљивих функција једнак скупу свих RM-израчунљивих, одн. скупу свих парцијално рекурзивних функција. Иако су while-програми доста 'чиљивији' од RM-програма, у овој књизи смо дали предност овој другој врсти програма јер су погоднији за теоријска разматрања о којима ће касније бити речи.

Докази теорема 4 и 5 показују да су примитивна рекурзија и минимизација дефинисане веома сличним дијаграмима тока, али и да постоји једна суштинска разлика међу њима: број итерација је фиксиран у случају примитивне рекурзије, док је непознат у случају минимизације. Међу свим `while`-програмима издвојићемо оне који генеришу примитивно рекурзивне функције.

for-програме дефинишемо индуктивно на следећи начин:

- $\mathsf{R}_k := \mathsf{R}_k + 1$ је for-програм дубине 0;
- $\mathsf{R}_k := \mathsf{R}_k \dot{-} 1$ је for-програм дубине 0;
- ако су $\mathbb{F}_1$ и $\mathbb{F}_2$ for-програми редом дубина n_1 и n_2, онда је и $(\mathbb{F}_1; \mathbb{F}_2)$ for-програм дубине $\max\{n_1, n_2\}$;

- ако је $\mathbb{F}$ for-програм дубине n и R_k регистар који се не помиње у $\mathbb{F}$, онда је $(\mathsf{for}\ \mathsf{R}_k\ \mathsf{do}\ \mathbb{F})$ for-програм дубине $n+1$.

Програм $(\mathsf{for}\ \mathsf{R}_k\ \mathsf{do}\ \mathbb{F})$ значи: „R_k пута понови $\mathbb{F}$“ и еквивалентан је следећем while-програму:

$$(\mathsf{while}\ \mathsf{R}_k \neq 0\ \mathsf{do}\ (\mathbb{F}; \mathsf{R}_k := \mathsf{R}_k \dot{-} 1)).$$

Теорема 12. *Функција је* for*-израчунљива акко је примитивно рекурзивна.*

Скица доказа. Наводимо само доказ чињенице да је свака for-израчунљива функција примитивно рекурзивна. Доказ се спроводи математичком индукцијом по дубини for-програма.

Ако је дубина for-програма нула (тј. ако нема for-наредби) програм се састоји од низа инструкција доделе, па је очигледно да свака променљива програма на крају извршавања има вредност коју одређује примитивно рекурзивна функција улазних променљивих.

Индуктивна претпоставка: претпоставимо да је тврђење тачно за све for-програме дубине n.

Размотрићемо случај када је $\mathbb{F}$ облика $(\mathsf{for}\ \mathsf{R}_{k+1}\ \mathsf{do}\ \mathbb{F}')$, за неки for-програм $\mathbb{F}'$ дубине n у коме се помињу само регистри $\mathsf{R}_1, \ldots, \mathsf{R}_k$.

Према индуктивној претпоставци закључујемо да постоје примитивно рекурзивне функције $g_1, \ldots, g_k$ такве да: ако су $x_1, \ldots, x_k$ редом садржаји регистара $\mathsf{R}_1, \ldots, \mathsf{R}_k$, онда је након једног извршавања програма $\mathbb{F}'$ за почетну конфигурацију $(x_1, \ldots, x_k)$ садржај регистра R_i једнак $g_i(x_1, \ldots, x_k)$. За свако $i \in \{1, \ldots, k\}$ нека је $f_i(x_1, \ldots, x_n, y)$ садржај регистра R_i након извршавања програма $\mathbb{F}'$ тачно y пута. Тада је

$$\begin{aligned}
&f_1(x_1, \ldots, x_k, 0) = x_1,\\
&\vdots\\
&f_k(x_1, \ldots, x_k, 0) = x_k,\\
&f_1(x_1, \ldots, x_k, y+1) = g_1(f_1(x_1, \ldots, x_k, y), \ldots, f_k(x_1, \ldots, x_k, y)),\\
&\vdots\\
&f_k(x_1, \ldots, x_k, y+1) = g_k(f_1(x_1, \ldots, x_k, y), \ldots, f_k(x_1, \ldots, x_k, y)).
\end{aligned}$$

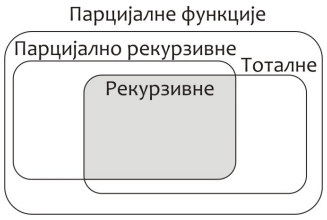
Како је скуп примитивно рекурзивних функција затворен за дефиниције симултаном рекурзијом (Теорема 8, страна 53), тврђење је доказано. □

Рекурзивне функције и рекурзивни скупови

Посебно важан подскуп скупа свих парцијално рекурзивних функција чине оне које су тоталне.

Дефиниција 9. ***Опште рекурзивне*** *(или само* ***рекурзивне****) функције јесу тоталне парцијално рекурзивне функције.*

Наравно, све примитивно рекурзивне функције јесу рекурзивне. Обрнуто није тачно.



Пример 34. **Акерманова функција** јесте пример рекурзивне функције која није примитивно рекурзивна. Ову функцију ћемо дефинисати полазећи од низа функција $a_0, a_1, a_2, \ldots$ датих са:

$$a_0(n) = n+1; \quad \left| \begin{array}{l} a_{m+1}(0) = a_m(1), \\ a_{m+1}(n+1) = a_m(a_{m+1}(n)). \end{array} \right.$$

Математичком индукцијом по m се доказује да је свака функција a_m примитивно рекурзивна. Уведене функције спадају у тзв. брзо растуће функције – толико брзе да за сваку примитивно рекурзивну функцију постоји нека функција a_m која је одозго ограничава. Наиме, важи следећа

Заправо, a_{m+1} се добија итерацијом функције a_m: $a_{m+1}(n) = a_m^{(n+1)}(1)$.

Теорема. *За сваку примитивно рекурзивну функцију* $f : \mathbb{N}^k \to \mathbb{N}$ *постоји природан број* ℓ *такав да је* $f(\vec{x}) < a_\ell(\max\{x_1, \ldots, x_k\})$. Идеја доказа је скицирана на маргини.[120]

Акерманова функција јесте бинарна функција $(m,n) \mapsto a_m(n)$, тј. функција A дата једнакостима:

$$\left| \begin{array}{l} A(0,n) = n+1, \\ A(m+1,0) = A(m,1) \\ A(m+1,n+1) = A(m, A(m+1,n)) \end{array} \right.$$

Ова функција је тотална и на јединствен начин одређена наведеним једнакостима. Интуитивно је јасно да је и парцијално рекурзивна, што ћемо доказати касније (страна 92). За сада само наводимо да се у сваком кораку израчунавања вредности $A(m,n)$ користи коначно много вредности функције A за *лексикографски мање* аргументе од (m,n). Међутим, функција A **није примитивно рекурзивна**. Ако би била, онда би и функција $A'(x) \stackrel{\text{def}}{=} A(x,x)$, $x \in \mathbb{N}$, била примитивно рекурзивна, па би постојао природан број $\ell \in \mathbb{N}$ (према Теореми наведеној у овом примеру) такав да је:

$$A'(x) < a_\ell(x) = A(\ell, x), \text{ за све } x \in \mathbb{N}.$$

Специјално, било би и $A(\ell,\ell) = A'(\ell) < a_\ell(\ell) = A(\ell,\ell)$, што није могуће.

Скуп рекурзивних функција може се увести аналогно Дефиницији 6 (којом се уводи скуп парцијално рекурзивних функција) уз

[120] Кажемо да функција $F : \mathbb{N} \to \mathbb{N}$ (строго) *ограничава* k-арну функцију $f : \mathbb{N}^k \to \mathbb{N}$ ако за све $\vec{x} \in \mathbb{N}^k$ важи $f(\vec{x}) < F(\max\{x_1, \ldots, x_k\})$. Наведена теорема је последица следећих чињеница:
(i) a_1 ограничава основне функције;
(ii) Ако функције $g_1, \ldots, g_m$ (дужине k) ограничавају редом $a_{\ell_1}, \ldots, a_{\ell_m}$, а функцију h (дужине m) ограничава a_{ℓ_0}, онда $\mathrm{Sup}_m^k(h; g_1, \ldots, g_m)$ ограничава a_ℓ, при чему је $\ell = \max\{\ell_0, \ell_1, \ldots, \ell_m\}$;
(iii) Ако функцију g (дужине k) ограничава a_{ℓ_0}, а функцију h (дужине $k+2$) ограничава a_{ℓ_1}, онда $\mathrm{Rec}^{k+1}(g,h)$ ограничава a_ℓ, при чему је $\ell = \max\{\ell_0, \ell_1, 1\}$.

рестрикцију на примену оператора неограничене минимизације. Рестрикцију оператора Min^k на тоталне функције $g : \mathbb{N}^{k+1} \to \mathbb{N}$ које задовољавају услов:

$$(\star) \qquad \forall \vec{x} \exists y (g(\vec{x}, y) = 0),$$

Формула $(\star)$ значи да једначина $g(\vec{x}, y) = 0$ има решења по y за сваки избор параметара $\vec{x}$.

називамо μ-рекурзијом. Кажемо да је функција

$$f(\vec{x}) = \mu y (g(\vec{x}, y) = 0)$$

добијена μ-**рекурзијом** функције g. Због услова $(\star)$, функција f је тотална.

Теорема 13. *Скуп рекурзивних функција је најмањи скуп функција који садржи основне функције и затворен је за супституцију, примитивну рекурзију и μ-рекурзију.*

Доказ. Нека је $\mathcal{C}$ скуп свих тоталних функција које задовољаву наведене услове. Истичемо да затвореност за μ-рекурзију значи:

- Ако $\mathcal{C}$ садржи функцију $g : \mathbb{N}^{k+1} \to \mathbb{N}$ за коју важи

  $$\forall \vec{x} \exists y (g(\vec{x}, y) = 0),$$

  онда и функција $f(x) = \mu y (g(\vec{x}, y) = 0)$ припада $\mathcal{C}$.

Очигледно, све функције из $\mathcal{C}$ јесу рекурзивне.

Обрнуто доказујемо ослањајући се на доказ Теореме 11. Нека је f било која k-арна тотална парцијално рекурзивна функција. Тада постоји програм $\mathbb{P}$ који израчунава њене вредности. Будући да је f тотална, за све $\vec{x} \in \mathbb{N}^k$, важи $\mathbb{P}(\vec{x}) \downarrow$. Према доказу Теореме 11, израчунавање програма $\mathbb{P}$ може се описати примитивно рекурзивном функцијом $\mathrm{Comp}_{\mathbb{P}} : \mathbb{N}^{k+1} \to \mathbb{N}$. Како се $\mathbb{P}$ зауставља за сваки улаз $\vec{x} \in \mathbb{N}$, закључујемо да важи:

$$(\star) \qquad \forall \vec{x} \exists t \, (\mathrm{Comp}_{\mathbb{P}}(\vec{x}, t))_0 = 0,$$

одакле следи да функција

$$\mathrm{Halt}_{\mathbb{P}}(\vec{x}) = \mu t \left((\mathrm{Comp}_{\mathbb{P}}(\vec{x}, t))_0 = 0 \right)$$

припада $\mathcal{C}$. Најзад, из

$$f(\vec{x}) = (\mathrm{Comp}_{\mathbb{P}}(\vec{x}, \mathrm{Halt}_{\mathbb{P}}(\vec{x})))_1$$

следи да и f припада $\mathcal{C}$. □

Дефиниција 10. *Скуп $R \subseteq \mathbb{N}^k$ је **рекурзиван** ако је рекурзивна карактеристична функција $\chi_R : \mathbb{N}^k \to \mathbb{N}$,*

$$\chi_R(\vec{x}) = \begin{cases} 1, & \vec{x} \in R, \\ 0, & \vec{x} \notin R. \end{cases}$$

Теорема 14. *(1)* ***Комплемент*** *рекурзивног скупа је рекурзиван скуп: ако је $R \subseteq \mathbb{N}^k$ рекурзиван скуп, онда је рекурзиван и скуп $R^{\complement}$.*

(2) ***Унија и пресек*** *рекурзивних скупова (исте дужине) јесу рекурзивни скупови: ако су $P, Q \subseteq \mathbb{N}^k$ рекурзивни скупови, онда су рекурзивни и скупови $P \cap Q$ и $P \cup Q$.*

Оператор μ-рекурзије можемо примењивати само на рекурзивне релације $R \subseteq \mathbb{N}^{k+1}$ које задовољавју услов $\forall \vec{x} \exists y\, R(\vec{x}, y)$, јер у том случају добијамо рекурзивну функцију $f(\vec{x}) = \mu y\, R(\vec{x}, y)$.

Теорема 15. *Нека је f произвољна тотална функција. Функција f је рекурзивна акко је њен график рекурзиван скуп.*

График k-арне функције f (која не мора бити тотална) јесте скуп

$$\Gamma_f = \{(\vec{x}, y) \mid f(\vec{x}) = y\} \subseteq \mathbb{N}^{k+1}.$$

Доказ. ($\Rightarrow$) Ако је f рекурзивна k-арна функција, таква је и функција

$$\chi_{\Gamma_f}(\vec{x}, y) = \begin{cases} 1, & f(\vec{x}) = y, \\ 0, & f(\vec{x}) \neq y. \end{cases}$$

($\Leftarrow$) Претпоставимо да је рекурзиван график Γ_f тоталне функције f. Будући да важи $\forall \vec{x} \exists y\, \Gamma_f(\vec{x}, y)$ (због претпоставке да је f тотална) и $f(\vec{x}) = \mu y \Gamma_f(\vec{x}, y)$, следи да је f рекурзивна функција. □

Пример 35. У претходној Теореми не може се изоставити претпоставка да је f тотална функција. Наиме, постоје парцијално рекурзивне функције, које нису тоталне, али имају рекурзиван график.

Уопште, k-арна парцијална функција φ има рекурзиван график акко постоји рекурзивна релација $R \subseteq \mathbb{N}^{k+1}$ таква да је $\varphi(\vec{x}) \simeq \mu y R(\vec{x}, y)$. Доказаћемо само нетривијални део ове еквиваленције. Нека је $R \subseteq \mathbb{N}^{k+1}$ рекурзивна релација и $\varphi(\vec{x}) \simeq \mu y R(\vec{x}, y)$. Рекурзивност графика Γ_φ следи из еквиваленције:

$$(\vec{x}, y) \in \Gamma_\varphi \;\Leftrightarrow\; R(\vec{x}, y) \wedge (\forall z < y)\, \neg R(\vec{x}, z).$$

Ако се програм $\mathbb{P}$ не зауставља за сваки улаз, функција $\mathrm{Halt}_{\mathbb{P}}$ (из доказа Теореме 11),

$$\mathrm{Halt}_{\mathbb{P}}(\vec{x}) = \mu t \left((\mathrm{Comp}_{\mathbb{P}}(\vec{x}, t))_0 = 0 \right),$$

јесте парцијално рекурзивна функција, која није тотална, али јој график јесте рекурзиван, будући да је релација '$(\mathrm{Comp}_{\mathbb{P}}(\vec{x}, t))_0 = 0$' рекурзивна.

Задаци

14. Да ли постоји рекурзивна функција чији је график примитивно рекурзиван, али она сама није примитивно рекурзивна?

Клинијева теорема о нормалној форми

Парцијална функција f је *интуитивно израчунљива* ако постоји „коначан опис поступка израчунавања“ вредности те функције. Према претходним резултатима, постоје различити, али међусобно еквивалентни приступи прецизирању „коначних описа поступка израчунавања“ (то могу бити Тјурингове машине, RM-програми, комбинатори, итд.) Најважнија чињеница својствена сваком од тих приступа јесте постојање **општег (универзалног) поступка** реализације било ког „коначног описа“ за задати улаз.

У случају RM-програма, тај **универзални поступак** је нефомално описан у претходном поглављу, и дефинисан је униформно за све RM-програме. Универзалним поступком се за сваки задати RM-програм $\mathbb{P}$ и сваки улаз $\overline{x}$ генерише израчунавање, тј. низ конфигурација $C_0, C_1, C_2, \ldots$, при чему је C_0 одређена улазом $\overline{x}$, а свака наредна конфигурација је потпуно одређена функцијом $\mathrm{NEXT}_{\mathbb{P}}$. Поред тога, наведен је и поступак којим се за сваку конфигурацију може утврдити да ли је завршна за програм $\mathbb{P}$ или није.

Кључна чињеница на којој ће бити базирано читаво ово поглавље јесте да се **универзални поступак** може описати једном парцијално рекурзивном функцијом, када природним бројевима искодирамо све што је релевантно за RM-програме и њихово извршавање. Другим речима, аритметизоваћемо RM-програме и њихова израчунавања, тј. читаву причу о RM-програмима испричаћемо језиком аритметике.

Кодирање инструкција RM-програма. Пошто су програми коначни низови инструкција, најпре ћемо све инструкције кодирати природним бројевима. Да бисмо донекле поједноставили разматрања, дозволићемо да параметар прелаза у инструкцијама буде и 0, при чему 0-та инструкција не постоји ни у једном програму. Дефинишимо бијекцију $[\cdot]$ која свакој RM-инструкцији додељује природан број – кôд те инструкције:

$[R_k^+ \mid \ell] = \langle 2(k-1), \ell\rangle, k \geqslant 1, \ell \geqslant 0$, и
$[R_k^- \mid \ell_1, \ell_2] = \langle 2(k-1)+1, \langle \ell_1, \ell_2\rangle\rangle, k \geqslant 1, \ell_1, \ell_2 \geqslant 0.$

$\langle x_1, x_2\rangle = 2^{x_1}(2x_2+1) - 1$
$\langle n\rangle_1 = (n+1)_0, \langle n\rangle_2 = \left[\frac{\frac{n+1}{2^{(n+1)_0}} - 1}{2}\right]$

Пример 36. Нађимо кодове неких инструкција:

$$[R_1^+ \mid 2] = \langle 2(1-1), 2\rangle = \langle 0, 2\rangle = 2^0 \cdot (2\cdot 2 + 1) - 1 = 4$$

$$[R_2^- \mid 2, 3] = \langle 2(2-1)+1, \langle 2, 3\rangle\rangle = \langle 3, \langle 2, 3\rangle\rangle$$
$$= \langle 3, 2^2(2\cdot 3 + 1) - 1\rangle = \langle 3, 27\rangle = 2^3(2\cdot 27 + 1) - 1 = 439$$

Одредимо инструкцију чији је кôд 12. Како је[123] $12 = \langle 0, 6\rangle$ и $0 = 2\cdot(1-1)$ је паран број, закључујемо да је $12 = [R_1^+ \mid 6]$.

Одредимо инструкцију чији је кôд 17. Како је[124] $17 = \langle 1, 4\rangle$ и $1 = 2(1-1)+1$ је непаран број, потребно је одредити ℓ_1, ℓ_2 такве да је $\langle \ell_1, \ell_2\rangle = 4$. Из[125] $4 = \langle 0, 2\rangle$, закључујемо да је $17 = [R_1^- \mid 0, 2]$.

[123] Решавањем једначине $12 = \langle x, y\rangle$, тј. $12 = 2^x(2y+1) - 1$ добијамо $x = 0, y = 6$.

[124] Решавањем једначине $17 = \langle x, y\rangle$, тј. $17 = 2^x(2y+1) - 1$ добијамо $x = 1, y = 4$.

[125] Решавањем једначине $4 = \langle \ell_1, \ell_2\rangle$, тј. $4 = 2^{\ell_1}(2\ell_2 + 1) - 1$ добијамо $\ell_1 = 0$, $\ell_2 = 2$.

За сваки природан број n, једноставно је наћи инструкцију чији је кôд n. Декодирање $[\cdot]^{-1}$, тј. поступак којим се за дати природни број одређује инструкција чији је кôд тај број, одређено је следећим једнакостима:

$$[n]^{-1} = \begin{cases} R^+_{\left[\frac{\langle n\rangle_1}{2}\right]+1} \mid \langle n\rangle_2, & \text{ако } 2 \mid \langle n\rangle_1, \\ R^-_{\left[\frac{\langle n\rangle_1}{2}\right]+1} \mid \langle\langle n\rangle_2\rangle_1, \langle\langle n\rangle_2\rangle_2, & \text{ако } 2 \nmid \langle n\rangle_1. \end{cases}$$

Кодирање RM-програма. Користећи уведено кодирање RM-инструкција, сваком RM-програму придружујемо кôд низа који чине кôдови инструкција тог програма: програму $\mathbb{P} = I_1, I_2, \ldots, I_m$ додељујемо кôд

$$[\![\mathbb{P}]\!] = \ulcorner [I_1], [I_2] \ldots, [I_m] \urcorner = 2^{[I_1]} + 2^{[I_1]+[I_2]+1} + \ldots + 2^{[I_1]+\cdots+[I_m]+m-1}.$$

Пример 37. Одредимо кôд програма:

$$\mathbb{P} : \begin{cases} 1. & R^-_2 \mid 2, 0 \\ 2. & R^+_1 \mid 1 \end{cases}$$

Како је[126] $[R^-_2 \mid 2, 0] = 55$ и $[R^+_1 \mid 1] = 2$, добијамо да је

$$[\![\mathbb{P}]\!] = \ulcorner 55, 2 \urcorner = 2^{55} + 2^{55+2+1} = 324\,259\,173\,170\,675\,712.$$

[126] $[R^-_2 \mid 2, 0] = \langle 2(2-1)+1, \langle 2, 0\rangle\rangle = 55$
$[R^+_1 \mid 1] = \langle 2(1-1), 1\rangle = 2$

Приметимо да кôд програма који има само једну инструкцију није једнак кôду те инструкције. На пример, ако је $\mathbb{P}'$ програм који има само једну инструкцију $1.R^-_1 \mid 0, 1$, онда је

$$[\![\mathbb{P}']\!] = \ulcorner [R^-_1 \mid 0, 1] \urcorner = \ulcorner 9 \urcorner = 2^9 = 512.$$

Функција $[\![\cdot]\!]$ јесте бијекција између скупа свих RM-програма и скупа природних бројева. За било који природан број n, једноставно одређујемо програм чији је кôд једнак n: најпре треба одредити низ чији је кôд једнак n, а затим одредити инструкције чији су кодови чланови тог низа. Прецизније, 0 је кôд празног програма[127], док је $n > 0$ кôд програма:

$$[\![n]\!]^{-1} = \begin{cases} 1. & [\lfloor n\rfloor_1]^{-1} \\ & \vdots \\ \mathrm{lh}(n). & [\lfloor n\rfloor_{\mathrm{lh}(n)}]^{-1} \end{cases}$$

[127] Празан програм је програм без инструкција. Свака почетна конфигурација је уједно и завршна конфигурација тог програма.

Уведено кодирање програма даје природним бројевима још једно („скривено“) значење – они представљају кодове програма.

Пример 38. Одредимо програм чији је кôд 178. Како је $178 = \ulcorner 1, 2, 0, 1 \urcorner$ (видети Пример 1, на страни 11), следи да је:

$178 = [10110010]_2 = \ulcorner 1, 2, 0, 1 \urcorner$

$$[\![178]\!]^{-1} = \begin{cases} 1. & [1]^{-1} \\ 2. & [2]^{-1} \\ 3. & [0]^{-1} \\ 4. & [1]^{-1} \end{cases} = \begin{cases} 1. & R^-_1 \mid 0, 0 \\ 2. & R^+_1 \mid 1 \\ 3. & R^+_1 \mid 0 \\ 4. & R^-_1 \mid 0, 0 \end{cases}$$

Ослањајући се на уведено (обострано-једнозначно) кодирање RM-програма, за $k \geqslant 1$ и $e \in \mathbb{N}$ нека је $\varphi_e^{(k)}$ k-арна парцијална функција коју израчунава програм чији је кôд e, тј.

$$\varphi_e^{(k)}(\vec{x}) \simeq \varphi_{[\![e]\!]^{-1}}^{(k)}(\vec{x}),\ \vec{x} \in \mathbb{N}^k.$$

Пример 39. Број 16 је кôд програма[129] (који има само једну инструкцију) $1.\ R_1^+ \mid 2$, па је $\varphi_{16}^{(k)}(x_1, \ldots, x_k) = x_1 + 1$.

[129] $16 = \lceil 4 \rceil = \lceil [R_1^+ \mid 1] \rceil$

За сваку k-арну парцијално рекурзивну функцију f постоји RM-програм $\mathbb{P}$ који је израчунава, одакле следи да се функција f сигурно налази негде у низу:

$$(*) \qquad \varphi_0^{(k)}, \varphi_1^{(k)}, \varphi_2^{(k)}, \varphi_3^{(k)}, \varphi_4^{(k)}, \ldots$$

Зато се низ $(*)$ назива и *нумерација* (набрајање) свих k-арних парцијално рекурзивних функција. Природан број e такав да је $f = \varphi_e^{(k)}$ називамо *индексом* функције f при нумерацији $(*)$. У нашем случају, индекс парцијално рекурзивне функције јесте кôд неког RM-програма који је израчунава. Свака парцијално рекурзивна функција има бесконачно много индекса.

Пример 40. Програм $1.\ R_1^- \mid 0, 1$ израчунава нула-функцију $\mathbf{0} : \mathbb{N} \to \mathbb{N}$, одакле следи да је $512 = [\![1.\ R_1^- \mid 0, 1]\!]$ један индекс функције $\mathbf{0}$. Међутим, индекси ове функције јесу и кодови следећих програма:

$$\left\{ \begin{array}{lll} 1. & R_1^- & \mid 0, 1 \\ 2. & R_1^+ & \mid 0 \end{array} \right. \quad \left\{ \begin{array}{lll} 1. & R_1^- & \mid 0, 1 \\ 2. & R_1^+ & \mid 0 \\ 3. & R_1^+ & \mid 0 \end{array} \right. \quad \left\{ \begin{array}{lll} 1. & R_1^- & \mid 0, 1 \\ 2. & R_1^+ & \mid 0 \\ 3. & R_1^+ & \mid 0 \\ 4. & R_1^+ & \mid 0 \end{array} \right. \quad \ldots$$

тј. бројеви 1536, 3584, 7680, ...

$[R_1^- \mid 1, 0] = 9$, $[R_1^+ \mid 0] = 0$,
$\lceil 9, 0 \rceil = 2^9 + 2^{9+0+1} = 1536$
$\lceil 9, 0, 0 \rceil = 2^9 + 2^{9+0+1} + 2^{9+0+0+2} = 3584$
$\lceil 9, 0, 0, 0 \rceil = 7680$

Кодирање конфигурација и функција next. Свака конфигурација регистар машине се може посматрати као низ природних бројева који су почев од неког члана сви једнаки нули. Конфигурације кодирамо природним бројевима већим од нуле:

$$\mathsf{C}: \boxed{i} \boxed{r_1} \boxed{\cdots} \boxed{r_d} \boxed{0 \cdots}$$

$$[\mathsf{C}] = p_0^i p_1^{r_1} \cdots p_d^{r_d}$$

Кодирање RM-инструкција и кодирање конфигурација означавамо истом ознаком $[\cdot]$, верујући да оваква злоупотреба не може довести до забуне; из контекста ће увек бити јасно шта је кодирано.

Уведено кодирање конфигурација представља бијекцију између скупа свих конфигурација и скупа $\mathbb{N}^+$. Свакој конфигурацији придружујемо јединствени кôд, и сваки природан број различит од 0 јесте кôд једне конфигурације коју једноставно добијамо растављањем тог броја на просте чиниоце.

Универзални поступак заправо је заснован на функцији NEXT чији је први аргумент RM-програм, а други нека конфигурација: за сваки RM-програм $\mathbb{P}$ и сваку конфигурацију C,

$$\mathsf{NEXT}(\mathbb{P},\mathsf{C}) = \text{ конфигурација која се добија из } \mathsf{C} \text{ извршавањем одговарајуће инструкције програма } \mathbb{P}.$$

Заправо, NEXT дефинишемо користећи функције $\mathrm{NEXT}_{\mathbb{P}}$, „подизањем“ програма $\mathbb{P}$ у аргумент. Разматрања испред Леме 5 (страна 58) указују на једнообразност аритметизације свих (за било које $\mathbb{P}$) функција $\mathrm{NEXT}_{\mathbb{P}}$. То запажање доводи до аритметичке варијанте функције NEXT, тј. до функције $\mathrm{next} : \mathbb{N}\times\mathbb{N}\to\mathbb{N}$, такве да за сваки програм $\mathbb{P}$ и сваку конфигурацију C важи једнакост $\mathrm{next}(\llbracket\mathbb{P}\rrbracket,\llbracket\mathsf{C}\rrbracket) = \llbracket\mathsf{NEXT}(\mathbb{P},\mathsf{C})\rrbracket$, односно

$$\mathrm{next}(e,c) = \text{ кôд наследника конфигурације чији је кôд } c \text{ током извршавања програма чији је кôд } e.$$

Очигледно је да постоји неформалан општи поступак којим за све природне бројеве e и c рачунамо вредност $\mathrm{next}(e,c)$, и то без икакве неограничене претраге, па је природно очекивати да је next примитивно рекурзивна функција.

Лема 6. *Постоји примитивно рекурзивна функција* $\mathrm{next}:\mathbb{N}^2\to\mathbb{N}$ *таква да за сваки програм* $\mathbb{P}$ *и сваку конфигурацију* C *важи једнакост* $\mathrm{next}(\llbracket\mathbb{P}\rrbracket,\llbracket\mathsf{C}\rrbracket) = \llbracket\mathsf{NEXT}(\mathbb{P},\mathsf{C})\rrbracket$.

Доказ.[132] Вредност $\mathrm{next}(e,c)$ рачунамо следећим поступком:

- Израчунај[133] $(c)_0$.
  - Ако[134] је $(c)_0 = 0$ или $(c_0) > \mathrm{lh}(e)$, онда је $\mathrm{next}(e,c) = \frac{c}{p_0^{(c)_0}}$;
  - Ако је $1 \leqslant (c)_0 \leqslant \mathrm{lh}(e)$, одреди[135] $\lfloor e\rfloor_{(c)_0}$:
    - Ако $2 \mid \langle\lfloor e\rfloor_{(c)_0}\rangle_1$, онда је $\lfloor e\rfloor_{(c)_0}$ кôд инструкције

      $$R^{+}_{\left\lceil\frac{\langle\lfloor e\rfloor_{(c)_0}\rangle_1}{2}\right\rceil+1} \mid \langle\lfloor e\rfloor_{(c)_0}\rangle_2,$$

      па је

      $$\mathrm{next}(e,c) = \frac{c}{p_0^{(c)_0}}\cdot p_0^{\langle\lfloor e\rfloor_{(c)_0}\rangle_2}\cdot p_{\left\lceil\frac{\langle\lfloor e\rfloor_{(c)_0}\rangle_1}{2}\right\rceil+1},$$

    - Ако $2 \nmid \langle\lfloor e\rfloor_{(c)_0}\rangle_1$, онда је $\lfloor e\rfloor_{(c)_0}$ кôд инструкције

      $$R^{-}_{\left\lceil\frac{\langle\lfloor e\rfloor_{(c)_0}\rangle_1}{2}\right\rceil+1} \mid \langle\langle\lfloor e\rfloor_{(c)_0}\rangle_2\rangle_1, \langle\langle\lfloor e\rfloor_{(c)_0}\rangle_2\rangle_2,$$

      одакле следи:

[132] Доказ служи само да строго потврди интуитивно јасну тврдњу да је next, са наведеним неформалним значењем, примитивно рекурзивна функција, и нема разлога да се трајно памте технички детаљи доказа.

[133] Израчунај редни број инструкције програма „e“ коју треба применити на c.

[134] $(c)_0$-та инструкција програма „e“ не постоји

[135] одреди кôд $(c)_0$-те инструкције програма „e“ и у складу са њом измени кôд c)

· Ако је $(c)_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1} \neq 0$, онда је

$$\mathrm{next}(e,c) = \frac{c}{p_0^{(c)_0} \cdot p_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1}} \cdot p_0^{\langle \langle \lfloor e \rfloor_{(c)_0} \rangle_2 \rangle_1},$$

· Ако је $(c)_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1} = 0$, онда је

$$\mathrm{next}(e,c) = \frac{c}{p_0^{(c)_0}} \cdot p_0^{\langle \langle \lfloor e \rfloor_{(c)_0} \rangle_2 \rangle_2}.$$

Све у свему:

$$\mathrm{next}(e,c) = \begin{cases} \frac{c}{p_0^{(c)_0}} & (c)_0 = 0 \vee (c_0) > \mathrm{lh}(e), \\ \frac{c}{p_0^{(c)_0}} \cdot p_0^{\langle \lfloor e \rfloor_{(c)_0} \rangle_2} \cdot p_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1} & 1 \leqslant (c)_0 \leqslant \mathrm{lh}(e) \wedge 2 \mid \langle \lfloor e \rfloor_{(c)_0} \rangle_1, \\ \frac{c}{p_0^{(c)_0} \cdot p_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1}} \cdot p_0^{\langle \langle \lfloor e \rfloor_{(c)_0} \rangle_2 \rangle_1} & 1 \leqslant (c)_0 \leqslant \mathrm{lh}(e) \wedge 2 \nmid \langle \lfloor e \rfloor_{(c)_0} \rangle_1 \wedge (c)_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1} \neq 0, \\ \frac{c}{p_0^{(c)_0}} \cdot p_0^{\langle \langle \lfloor e \rfloor_{(c)_0} \rangle_2 \rangle_2} & 1 \leqslant (c)_0 \leqslant \mathrm{lh}(e) \wedge 2 \nmid \langle \lfloor e \rfloor_{(c)_0} \rangle_1 \wedge (c)_{\left\lceil \frac{\langle \lfloor e \rfloor_{(c)_0} \rangle_1}{2} \right\rceil + 1} = 0, \end{cases}$$

па је јасно да је $\mathrm{next} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ примитивно рекурзивна функција. □

Кодирање израчунавања и Клинијеви предикати. Преостаје још да се позабавимо коначним израчунавањима. Сваком коначном низу конфигурација $\mathcal{I} = \mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_n$ придружујемо кôд низа који чине кôдови појединих чланова: $[\![\mathcal{I}]\!] = \lceil [\mathsf{C}_1], [\mathsf{C}_2] \ldots, [\mathsf{C}_n] \rceil$.

Кључно питање јесте: Да ли неки коначан низ конфигурација $\mathsf{C}_1, \ldots, \mathsf{C}_n$ представља израчунавање програма $\mathbb{P}$ за улаз $\vec{x} \in \mathbb{N}^k$? Јасно је да постоји неформалан општи поступак који даје одговор на претходно питање:

- прво треба проверити да ли је C_1 одговарајућа почетна конфигурација за улаз $\vec{x}$;
- затим, да ли је $\mathsf{NEXT}(\mathbb{P}, \mathsf{C}_i) = \mathsf{C}_{i+1}, 1 \leqslant i < n$;
- најзад, да ли је C_n завршна конфигурација за програм $\mathbb{P}$.

Узимајући у обзир описана кодирања налазимо аритметичку варијанту овог поступка познату као **Клинијев предикат**:

$T_k(e,\vec{x},z) \overset{\text{def}}{\Leftrightarrow}$ z је кôд израчунавања програма чији је кôд e за улаз $\vec{x}$

$\Leftrightarrow$
- $\lfloor z\rfloor_1$ је кôд почетне конфигурације за улаз $\vec{x}$, тј.
  у бројачу је уписано 1, $(\lfloor z\rfloor_1)_0 = 1$,
  $\vec{x}$ је уписано у $\mathsf{R}_1,\ldots,\mathsf{R}_k$, $(\lfloor z\rfloor_1)_i = x_i, 1 \leqslant i \leqslant k$,
  а у осталим регистрима, који се помињу у програму e, уписана је 0, $\forall j \leqslant \mathrm{lh}(e)(1 \leqslant j \wedge \left[\frac{\langle\lfloor e\rfloor_j\rangle_1}{2}\right] \geqslant k \Rightarrow (\lfloor z\rfloor_1)_{\left[\frac{\langle\lfloor e\rfloor_j\rangle_1}{2}\right]+1} = 0)$
- за $1 \leqslant i < \mathrm{lh}(z)$, $\lfloor z\rfloor_{i+1}$ је кôд следбеника конфигурације $\lfloor z\rfloor_i$ током извршавања програма чији је кôд e, $\mathrm{next}(e,\lfloor z\rfloor_i) = \lfloor z\rfloor_{i+1}, 1 \leqslant i < \mathrm{lh}(z)$
- $\lfloor z\rfloor_{\mathrm{lh}(z)}$ је кôд завршне конфигурације програма чији је кôд e. $\left(\lfloor z\rfloor_{\mathrm{lh}(z)}\right)_0 > \mathrm{lh}(e) \vee \left(\lfloor z\rfloor_{\mathrm{lh}(z)}\right)_0 = 0$

Предикат $T_k(e,\vec{x},z)$ је конјункција свих формула наведених са десне стране, одакле следи да је примитивно рекурзиван.[136]

Будући да је садржај регистра R_1 у завршној конфигурацији резултат израчунавања по неком RM-програму за задати улаз, дефинишимо функцију $U : \mathbb{N} \to \mathbb{N}$ са следећим неформалним значењем:

$U(z) =$ садржај регистра R_1 у последњој конфигурацији
низа конфигурација чији је кôд z,

које преводимо у строгу дефиницију ослањајући се на уведена кодирања: $U(z) = \left(\lfloor z\rfloor_{\mathrm{lh}(z)}\right)_1$.

Теорема 16. [***Клинијева теорема о нормалној форми***] *Постоји примитивно рекурзивна функција $U : \mathbb{N} \to \mathbb{N}$ и за свако $k \geqslant 1$ примитивно рекурзивна $(k+2)$-арна релација T_k тако да за свако $\vec{x} \in \mathbb{N}^k$ важи:*

- $\vec{x} \in \mathrm{dom}(\varphi_e^{(k)}) \Leftrightarrow \exists z\, T_k(e,\vec{x},z)$,
- $\varphi_e^{(k)}(\vec{x}) \simeq U(\mu z\, T_k(e,\vec{x},z))$.

Клинијева теорема о нормалној форми значајно поједностављује проучавање скупа парцијално рекурзивних функција, па ћемо је доста пута примењивати у наредним одељцима.

Последица 6. [***Теорема о ефективности нумерације***] *За свако $k \geqslant 1$, парцијална $(k+1)$-арна функција*

$$(*) \qquad (e,\vec{x}) \mapsto U(\mu z\, T_k(e,\vec{x},z))$$

јесте парцијално рекурзивна.

[136] Приметимо да за било које $e \in \mathbb{N}$ и $\vec{x} \in \mathbb{N}^k$, ако постоји $z \in \mathbb{N}$ такав да важи $T_k(e,\vec{x},z)$, онда постоји чак бесконачно много бројева z' за које ће такође важити $T_k(e,\vec{x},z')$. То је последица чињенице да израчунавањем проглашавамо све коначне низове конфигурација који задовоље постављене услове само у односу на регистре који су релевантни током извршавања одговарајућег програма.

Функцију $(*)$ означавамо $\Phi_U^{(k)}$ и називамо **универзалном функцијом** за k-арне парцијално рекурзивне функције[137], јер је

$$\Phi_U^{(k)}(e, \vec{x}) \simeq \varphi_e^{(k)}(\vec{x}), \text{ за све } e \in \mathbb{N}, \vec{x} \in \mathbb{N}^k.$$

Другим речима, $\Phi_U^{(k)}(e, \vec{x})$ је резултат извршавања RM-програма чији је кôд e за улаз $\vec{x}$ ако се то израчунавање зауставља, а у супротном $\Phi_U^{(k)}(e, \vec{x})$ није дефинисано.

[137] Када је $k = 1$, горњи индекс изостављамо и уместо $\Phi_U^{(1)}$ пишемо Φ_U.

Пример 41. 1) Израчунајмо $\Phi_U^{(2)}(16, 21, 34)$. Према примеру 39, знамо да је 16 кôд програма $1.R_1^+ \mid 2$. Овај програм за улаз $(21, 34)$ даје резултат 22. Дакле, $\Phi_U^{(2)}(16, 21, 34) = 22$.

2) Израчунајмо $\Phi_U^{(4)}(178, 1, 2, 3, 3)$:

- Одредимо програм чији је кôд 178 (пример 38, страна 66):

$$[\![178]\!]^{-1} = \begin{cases} 1. & [1]^{-1} \\ 2. & [2]^{-1} \\ 3. & [0]^{-1} \\ 4. & [1]^{-1} \end{cases} = \begin{cases} 1. & R_1^- \mid 0, 0 \\ 2. & R_1^+ \mid 1 \\ 3. & R_1^+ \mid 0 \\ 4. & R_1^- \mid 0, 0 \end{cases}$$

- Израчунавање програма $[\![178]\!]^{-1}$ за улаз $(1, 2, 3, 3)$ је:

$$1; 1, 2, 3, 3 \to^*_{[\![178]\!]^{-1}} 0; 0, 2, 3, 3.$$

Дакле, $\Phi_U^{(4)}(178, 1, 2, 3, 3) = 0$.

Последица 6 је веома значајна, јер омогућава да приликом састављања RM-програма позивамо (као потпрограме) програме за универзалне функције.[138]

[138] Позивањем универзалних функција заправо захтевамо да се природан број (записан у неком регистру) третира као кôд програма и да се тај програм изврши за улаз који је одређен садржајима неких изабраних регистара. Другим речима, RM-програми могу да садрже наредбе облика $R_j := \Phi_U^k(R_i, R_{i_1}, \ldots, R_{i_k})$, чије је значење: изврши програм чији је кôд записан у R_i узимајући за улаз садржаје регистара $R_{i_1}, \ldots, R_{i_k}$ и резултат (у случају заустављања) упиши у R_j. Наравно, могуће је да неки од $i_1, \ldots, i_k$ буде једнак i.

Пример 42. Позивањем потпрограма за одговарајуће универзалне функције, није тешко саставити програм који на улазу добија три броја x_1, x_2, y и обавља следећи задатак: прва два третира као кодове RM-програма, оба програма извршава за улаз y, добијене резултате (ако их достигне) сабира и тај збир враћа као излаз. Описани програм заправо израчунава функцију: $f(x_1, x_2, y) \simeq \varphi_{x_1}(y) + \varphi_{x_2}(y)$. Генерално, у оваквим ситуацијама, за нашу причу је најважније да такав програм постоји, што једноставно закључујемо из чињенице да је f парцијално рекурзивна функција будући да је $f(x_1, x_2, y) \simeq \Phi_U(x_1, y) + \Phi_U(x_2, y)$.

Пример 43. Унарна функција $x \mapsto \varphi_x(x)$ јесте парцијално рекурзивна, будући да је $\varphi_x(x) \simeq \Phi_U(x, x)$ и $x \mapsto \Phi_U(x, x)$ је парцијално рекурзивна функција. RM-програм који израчунава ову функцију користи улаз x као кôд одговарајућег програма, али и као улаз за тај програм. Овакве ситуације нису необичне: ако (у неком програмском језику) саставимо програм S који испитује синтаксну коректност других програма (изабраног језика), онда програму S као улаз можемо задати сâм S.

ПРИМЕР 44. Бинарна функција дефинисана са:

$$f(x,y) \simeq \varphi_{x+y}(xy) + \varphi_{xy}(x+y),\ x,y \in \mathbb{N}$$

јесте парцијално рекурзивна, јер је

$$f(x,y) \simeq \Phi_U(x+y,xy) + \Phi_U(xy,x+y).$$

Резимирајмо укратко основне резултате овог одељка. За свако $k \geqslant 1$, све k-арне парцијално рекурзивне функције смо поређали у низ $(\varphi_e^{(k)})_{e\in\mathbb{N}}$ (сваки члан низа је парцијално рекурзивна функција и свака k-арна парцијално рекурзивна функција јесте члан низа). Овај низ називамо и *ефективним набрајањем* свих k-арних парцијално рекурзивних функција, будући да постоји $(k+1)$-арна парцијално рекурзивна функција Φ_U^k таква да је $\Phi_U^k(e,\vec{x}) \simeq \varphi_e(\vec{x})$, за све $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^k$. Наведени резултати заправо истичу нову улогу коју ће природни бројеви имати у наставку поглавља: природни бројеви, поред своје уобичајене улоге, преузимају и улогу програма; уместо са програмима најчешће ћемо радити са њиховим кодовима.

Универзални програми

Универзални RM-програм. Посматрање универзалних функција за сваку појединачну дужину k парцијално рекурзивних функција има својих предности, али у суштини није неопходно. Поред кодирања која само увели у претходном одељку, могли смо да кодирамо и све могуће улазе (произвољне дужине): сваком $\vec{x} \in \bigcup_{k\geqslant 1} \mathbb{N}^k$ да доделимо кôд $\lceil \vec{x} \rceil$. То би нас довело до примитивно рекурзивног предиката $T \subseteq \mathbb{N}^3$:

$$T(e,x,z) \Leftrightarrow z \text{ је кôд израчунавања програма чији је кôд } e \text{ за улаз чији је кôд } x.$$

Строга дефиниција предиката T аналогна је дефиницији предиката T_k, при чему треба додати услов $x \neq 0$, уместо k писати $\mathrm{lh}(x)$, а уместо x_i писати $\lfloor x \rfloor_i$:

$$\begin{aligned} T(e,x,z) \overset{\text{def}}{\Leftrightarrow}\ & x \neq 0 \wedge (\lfloor z \rfloor_1)_0 = 1 \wedge (\forall i \leqslant \mathrm{lh}(x))(i \geqslant 1 \Rightarrow (\lfloor z \rfloor_1)_i = \lfloor x \rfloor_i) \wedge \\ & \wedge \forall j \leqslant \mathrm{lh}(e)(1 \leqslant j \wedge \left[\frac{\langle \lfloor e \rfloor_j \rangle_1}{2}\right] \geqslant \mathrm{lh}(x) \Rightarrow (\lfloor z \rfloor_1)_{\left[\frac{\langle \lfloor e \rfloor_j \rangle_1}{2}\right]+1} = 0) \wedge \\ & \wedge (\forall i < \mathrm{lh}(z))(i \leqslant 1 \Rightarrow \mathrm{next}(e, \lfloor z \rfloor_i) = \lfloor z \rfloor_{i+1}) \wedge \\ & \wedge \left(\lfloor z \rfloor_{\mathrm{lh}(z)}\right)_0 > \mathrm{lh}(e) \vee \left(\lfloor z \rfloor_{\mathrm{lh}(z)}\right)_0 = 0. \end{aligned}$$

Очигледно, бинарна функција Φ дефинисана са

$$\Phi(e,x) \simeq U(\mu z T(e,x,z)),$$

јесте парцијално рекурзивна, па постоји RM-програм $\mathbb{U}$ који је израчунава. Овај програм називамо универзалним RM-програмом, јер $\mathbb{U}$ на одређени начин извршава све RM-програме за задате улазе: када $\mathbb{U}$ покренемо за „програм“ e и „улаз“ x као резултат добијамо заправо резултат извршавања „програма“ e за „улаз“ x. Другим речима, за сваки RM-програм $\mathbb{P}$ и сваки коначан низ природних бројева $\vec{x}$:

$$\mathbb{U}([\![\mathbb{P}]\!], \lceil \vec{x} \rceil) \uparrow \quad \text{акко} \quad \mathbb{P}(\vec{x}) \uparrow$$

и за све $y \in \mathbb{N}$,

$$\mathbb{U}([\![\mathbb{P}]\!], \lceil \vec{x} \rceil) \downarrow y \quad \text{акко} \quad \mathbb{P}(\vec{x}) \downarrow y.$$

Следећа интерпретација претходне приче од суштинског је значаја за рачунарство и конструкцију самог рачунара:

- универзални програм $\mathbb{U}$ репрезентује сам механизам који извршава RM-програме (слободније речено, хардвер);

- кодирање свих RM-програма представља језик на коме механизму саопштавамо RM-програме;
- кодирање улаза представља договорени начин задавања улаза RM-програмима.

Ово изједначавање хардвера, програма и њихових улаза истиче се као кључно запажање које је покренуло изградњу рачунара. До тог запажања је дошао, наравно, Тјуринг разматрајући машине које су по њему назване.

Универзална Тјурингова машина. Пре него што дефинишемо универзалну Тјурингову машину, неопходно је да све Тјурингове машине кодирамо речима неког унапред задатог алфабета. Показаћемо како се све Тјурингове машине могу кодирати речима алфабета $\{0,1\}$. Без губљења општости, претпоставићемо да је свако стање *било које* Тјурингове машине означено q_i, за неки природан број i, а да је сваки симбол траке означен s_i, за неки природан број i. Прецизније, за сваку Тјурингову машину $\mathbb{T} = (Q, q_0, F, \Gamma, \sqcup, \Sigma, \tau)$, могу се изабрати природни бројеви $m \geqslant 0$ и $n \geqslant 1$, и скупови $\{i_1, \ldots, i_k\} \subseteq \{0, \ldots, m\}$ и $\{j_1, \ldots, j_\ell\} \subseteq \{1, \ldots, n\}$ такви да је:

$$\begin{array}{lll} Q = \{q_0, \ldots, q_m\}, & & F = \{q_{i_1}, \ldots, q_{i_k}\}, \\ \Gamma = \{s_0, \ldots, s_n\}, & \sqcup = s_0, & \Sigma = \{s_{j_1}, \ldots, s_{j_\ell}\}, \end{array}.$$

Функцију прелаза τ идентификујемо са низом уређених петорки из $(Q \setminus F) \times \Gamma \times \Gamma \times \{\mathsf{R}, \mathsf{L}, \mathsf{P}\} \times Q$ таквих да за сваки уређени пар из $(Q \setminus F) \times \Gamma$, у низу τ постоји тачно једна петорка чије су прве две координате одређене тим паром.

Уз додатни договор да стање q_i идентификујемо са речју $q\mathrm{bin}(i)$, а симбол s_i са речју $s\mathrm{bin}(i)$,[139] закључујемо да сваку Тјурингову машину можемо задати као реч алфабета $\{0, 1, q, s, \mathsf{R}, \mathsf{L}, \mathsf{P}, ;\}$ која је следећег облика:

[139] $q_0 \equiv q0$, $q_1 \equiv q1$, $q_2 \equiv q10$, $q_3 \equiv q11$, $\ldots$
$s_0 \equiv s0$, $s_1 \equiv s1$, $s_2 \equiv s10$, $s_3 \equiv s11$, $\ldots$

$$(\star) \qquad q0 \cdots q\mathrm{bin}(m); q0; q\mathrm{bin}(i_1) \cdots q\mathrm{bin}(i_k); s0 \cdots s\mathrm{bin}(n); s0; s\mathrm{bin}(j_1) \cdots s\mathrm{bin}(j_\ell); \overline{\tau}$$

при чему је $\overline{\tau}$ реч добијена надовезивањем речи

$$q\mathrm{bin}(i)s\mathrm{bin}(j)s\mathrm{bin}(j')\mathsf{D}q\mathrm{bin}(i')$$

које су преводи одговарајућих петорки $q_i s_j s_{j'} \mathsf{D} q_{j'}$ низа τ.

Кодирањем симбола алфабета $\{0, 1, q, s, \mathsf{R}, \mathsf{L}, \mathsf{P}, ;\}$ трословним речима из Σ_{Bool}:

$$\begin{pmatrix} 0 & 1 & q & s & \mathsf{R} & \mathsf{L} & \mathsf{P} & ; \\ 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{pmatrix}$$

реч облика $(\star)$ преводимо у реч алфабета $\{0,1\}$ коју ћемо називати кôдом Тјурингове машине $\mathbb{T}$ и обележавати $[\![\mathbb{T}]\!]$. Очигледно је да

постоји (чак бесконачно много) речи над $\{0,1\}$ које нису кодови ниједне Тјурингове машине.[140] Да нема разлога за бригу потврђује следећа лема.

[140] То није био случај са кодирањем RM-програма, када је сваки природан број био кôд неког RM-програма.

Лема 7. *Језик*

$$\mathrm{Syntax} = \{t \in \{0,1\}^* \mid t \textit{ је кôд неке Тјурингове машине}\}$$

је рекурзиван.

Доказ претходне леме изостављамо, ослањајући се на чињеницу да се лако може уочити неформална процедура којом се проверава да ли је задата реч $t \in \{0,1\}^*$ кôд неке Тјурингиве машине или није. Иначе, суштину изостављеног доказа чини конструкција Тјурингове машине $\mathbb{S}$:

$$\mathbb{S} = (Q^{\mathbb{S}}, \mathsf{q}_0^{\mathbb{S}}, \{\mathsf{q}_{\mathrm{syntax-correct}}, \mathsf{q}_{\mathrm{syntax-error}}\}, \Gamma^{\mathbb{S}}, \sqcup, \Sigma_{\mathrm{Bool}}, \tau^{\mathbb{S}}\})$$

такве да за сваку реч $t \in \{0,1\}^*$ важи:

$$\mathsf{q}_0^{\mathbb{S}} t \to_{\mathbb{S}}^* \begin{cases} \mathsf{q}_{\mathrm{syntax-correct}} t, & t \text{ јесте кôд неке Тјурингове машине,} \\ \mathsf{q}_{\mathrm{syntax-error}} \sqcup, & t \text{ није кôд неке Тјурингове машине.} \end{cases}$$

На основу наведеног кодирања Тјурингових машина, јасно је како се кодирају коначни низови речи улазног алфабета: за улаз $\overline{w} = (w_1, \ldots, w_k) \in \Sigma^{*k}$ одговарајући почетни садржај траке

$$(\star) \qquad w_1 \sqcup w_2 \sqcup \cdots \sqcup w_k$$

описујемо наводећи редом кодове свих симбола који се појављују у $(\star)$ и на тај начин формирамо кôд улаза $[\![\overline{w}]\!] \in \{0,1\}^*$.

Лема 8. *Језик*

$$\mathrm{Input} = \{(t,w) \in \{0,1\}^{*2} \mid t \in \mathrm{Syntax} \textit{ и } w \textit{ је кôд коначног низа речи улазног алфабета Тјурингове машине чији је кôд } t\}$$

је рекурзиван.

Сваку реч C која може бити конфигурација неке Тјурингове машине кодирамо на очекивани начин; тај кôд такође означавамо $[\![\mathsf{C}]\!]$. Даље би се, наравно, након кодирања низова конфигурација, могао дефинисати и одговарајући Клинијев предикат, што ми нећемо чинити, већ прелазимо на кратак опис универзалног Тјуринговог програма.

Централно место у опису рада Тјурингових машина заузима поступак којим се обавља следећи задатак: за задату Тјурингову машину $\mathbb{T}$ и задат коначан низ $\overline{w}$ речи њеног улазног алфабета, генерисати конфигурације одговарајућег израчунавања, препознати завршну конфигурацију и зауставити се. Важно Тјурингово

запажање је да постоји **универзална Тјурингова машина** $\mathbb{U}$ која обавља овај задатак када јој се задају кôд неке Тјурингове машине и кôд одговарајућег улаза.

Да би прича била потпунија, можемо претпоставити да уколико $\mathbb{U}$ добије улаз (t,w) такав да t није кôд неке Тјурингове машине или, ако јесте, w није кôд коначног низа речи улазног алфабета, онда $\mathbb{U}$ штампа SYNTAX/INPUT ERROR. Наравно, овакав договор неће бити ни од каквог значаја у наставку.

Теорема 17. *Постоји Тјурингова машина* $\mathbb{U}$ *таква да за сваку Тјурингову машину* $\mathbb{T}$ *и коначан низ* $\overline{w}$ *речи њеног улазног алфабета важи:*

$$\mathbb{U}(\llbracket\mathbb{T}\rrbracket, \llbracket\overline{w}\rrbracket)\uparrow \quad \textit{акко} \quad \mathbb{T}(\overline{w})\uparrow;$$

штавише, ако $\mathbb{U}(\llbracket\mathbb{T}\rrbracket, \llbracket\overline{w}\rrbracket)\downarrow$*, онда је завршна конфигурација овог израчунавања једнака* $\mathsf{q}^{\mathbb{U}}_{\mathrm{stop}}\llbracket\mathsf{C}\rrbracket$*, где је* C *завршна конфигурација израчунавања машине* $\mathbb{T}$ *за улаз* $\overline{w}$*.*

Доказ ове теореме може се извести из постојања универзалног RM-програма и резултата одељка о односу Тјурингових машина и RM-програма.

Рекурзивно набројиви скупови

Пре него што смо дефинисали појам RM-израчунљиве функције (видети страну 38), истакли смо да сваки RM-програм $\mathbb{P}$ за свако $k \geqslant 1$ одређује по један подскуп од $\mathbb{N}^k$ који садржи све оне k-торке природних бројева за које се $\mathbb{P}$ зауставља:

$$W_{\mathbb{P}}^{(k)} = \{(x_1, \ldots, x_k) \in \mathbb{N}^k \mid \mathbb{P}(x_1, \ldots, x_k) \downarrow\}.$$

Сваки овакав скуп је домен одговарајуће RM-израчунљиве функције: $W_{\mathbb{P}}^{(k)} = \mathrm{dom}(\varphi_{\mathbb{P}}^{(k)})$.

Дефиниција 11. *Скуп $A \subseteq \mathbb{N}^k$ је **полуодлучив** или **рекурзивно набројив** (краће р. н.) ако је домен неке парцијално рекурзивне функције, тј. постоји $e \in \mathbb{N}$ такав да је $A = \mathrm{dom}(\varphi_e^{(k)})$.*

Према претходној дефиницији, полуодлучивост неког скупа A потврђује парцијално рекурзивна функција чији је домен тај скуп, при чему нису важне вредности функције за аргументе из A. Зато је згодно, полуодлучивост окарактерисати тзв. *парцијалним карактеристичним функцијама*, које донекле оправдавају зашто домене парцијално рекурзивних функција називамо *полуодлучивим.* [142]

[142] Неформално говорећи, парцијална карактеристична функција само потврђује припадност одговарајућем скупу, али не препознаје неприпадање.

Дефиниција 12. *Нека је $A \subseteq \mathbb{N}^k$. Парцијална карактеристична функција скупа A је k-арна функција $\overline{\chi}_A$ дефинисана са*

$$\overline{\chi}_A(\vec{x}) = \begin{cases} 1, & \vec{x} \in A, \\ \uparrow, & \vec{x} \notin A. \end{cases}$$

Лема 9. *Скуп $A \subseteq \mathbb{N}^k$ је полуодлучив (рекурзивно набројив) акко је k-арна парцијална функција $\overline{\chi}_A$ парцијално рекурзивна.*

Доказ. Ако је $A \subseteq \mathbb{N}^k$ полуодлучив, онда је $A = \mathrm{dom}(\varphi_e^{(k)})$, за неко $e \in \mathbb{N}$. Није тешко уочити да је $\overline{\chi}_A(\vec{x}) \simeq \mathbf{1}(\varphi_e^{(k)}(\vec{x}))$, одакле следи да је $\overline{\chi}_A$ парцијално рекурзивна функција.

Обрнуто, ако је $\overline{\chi}_A$ парцијално рекурзивна функција, онда је A полуодлучив, јер је $A = \mathrm{dom}(\overline{\chi}_A)$. □

Лема 10. *Сваки одлучив (тј. рекурзиван) скуп је полуодлучив (тј. рекурзивно набројив).*

Доказ. Скуп $A \subseteq \mathbb{N}^k$ је одлучив (рекурзиван) ако је парцијално рекурзивна његова карактеристична функција χ_A (која је тотална!):

$$\chi_A(\vec{x}) = \begin{cases} 1, & \vec{x} \in A, \\ 0, & \vec{x} \notin A. \end{cases}$$

Из парцијалне рекурзивности функције χ_A следи парцијална рекурзивност функције $\overline{\chi}_A$. Ово можемо показати, на пример, користећи парцијалну рекурзивност функције (која је попут оне из

Примера 28 на страни 45):

$$g(x) = \begin{cases} 1, & x = 0, \\ \uparrow, & x > 0. \end{cases}$$

Очигледно је $\overline{\chi}_A(\vec{x}) \simeq g(1 \dot{-} \chi_A(\vec{x}))$. □

Најпознатији пример полуодлучивог скупа који није рекурзиван назива се *комбинаторно језгро неодлучивости* и обележава се

$$K \stackrel{\text{def}}{=} \{e \mid \varphi_e(e) \downarrow\}.$$

Овај скуп ћемо још доста пута помињати.

Теорема 18. *Скуп K је полуодлучив, али није одлучив.*

Доказ. Да је K полуодлучив следи из чињенице да је он домен парцијално рекурзивне функције $x \mapsto \Phi_U(x, x)$.

Претпоставимо да K јесте рекурзиван, тј. да је рекурзивна функција $\chi_K : \mathbb{N} \to \mathbb{N}$,

$$\chi_K(x) = \begin{cases} 1, & \varphi_x(x) \downarrow, \\ 0, & \varphi_x(x) \uparrow. \end{cases}$$

Тада је унарна функција h,

$$h(x) = \begin{cases} \uparrow, & \varphi_x(x) \downarrow, \\ 0, & \varphi_x(x) \uparrow. \end{cases}$$

парцијално рекурзивна. Нека је e_0 индекс функције h: $\varphi_{e_0}(x) \simeq h(x)$, за свако $x \in \mathbb{N}$. Специјално, за $x = e_0$, $\varphi_{e_0}(e_0) \simeq h(e_0)$, а то није могуће, јер према дефиницији функције h, за било које x важи: $h(x) \downarrow$ акко $\varphi_x(x) \uparrow$.

Дакле, скуп K не може бити рекурзиван. □

Наредна теорема исказује једну од најзначајнијих особина рекурзивно набројивих скупова.

Једнакост $A = \{\vec{x} \mid \exists y R(\vec{x}, y)\}$ може се формулисати и у облику еквиваленције: $A(\vec{x}) \Leftrightarrow \exists y R(\vec{x}, y)$.
Према наведеној теореми, рекурзивно набројиви скупови су пројекције рекурзивних скупова.

Теорема 19. *Скуп $A \subseteq \mathbb{N}^k$ је полуодлучив (рекурзивно набројив) акко постоји рекурзиван скуп $R \subseteq \mathbb{N}^{k+1}$ такав да је*

$$A = \{\vec{x} \mid \exists y R(\vec{x}, y)\}.$$

Доказ. ($\Rightarrow$) Ако је $A \subseteq \mathbb{N}^k$ рекурзивно набројив, онда је, за неко $e_0 \in \mathbb{N}$, $A = \mathrm{dom}(\varphi^{(k)}_{e_0})$. Према Клинијевој теореми о нормалној форми, за све $\vec{x} \in \mathbb{N}^k$ важи:

$$A(\vec{x}) \Leftrightarrow \vec{x} \in \mathrm{dom}(\varphi^{(k)}_{e_0}) \Leftrightarrow \exists z T_k(e_0, \vec{x}, z).$$

Будући да је T_k примитивно рекурзивна релација, таква је и релација $R \subseteq \mathbb{N}^{k+1}$ добијена из T_k за изабрано e_0: $R(\vec{x}, z) \stackrel{\text{def}}{\Leftrightarrow} T_k(e_0, \vec{x}, z)$. Дакле, $A(\vec{x}) \Leftrightarrow \exists z\, R(\vec{x}, z)$.

($\Leftarrow$) Ако је $A(\vec{x}) \Leftrightarrow \exists y R(\vec{x}, y)$, где је $R \subseteq \mathbb{N}^{k+1}$ рекурзиван скуп, тада је A домен парцијално рекурзивне функције φ дефинисане са

$$\varphi(\vec{x}) \simeq \mu y R(\vec{x}, y),$$

одакле следи да је A рекурзивно набројив. □

Сликовито објашњење претходне теореме наводимо за унарне релације. За сваки полуодлучив скуп $A \subseteq \mathbb{N}$ постоји бинарна одлучива релација $R \subseteq \mathbb{N}^2$ таква да $A(x) \Leftrightarrow \exists y R(x, y)$. То значи да алгоритам којим се испитује припадање и неприпадање скупу R одређује следећи „полуалгоритам“ којим се испитује припадање, али не и неприпадање, скупу A: за дато $x \in \mathbb{N}$, редом испитуј да ли је или није,

$$R(x,0),\ R(x,1),\ R(x,2),\ldots$$

и заустави се одмах после потврдног одговора. Овај полуалгоритам се зауставља само у случају да $x \in A$. Али, ако $x \notin A$, одн. $\forall y\, \neg R(x, y)$, полуалгоритам „не види“ да не постоји y такав да је $R(x, y)$, већ наставља са радом након сваког негативног одговора: $\neg R(x,0),\ \neg R(x,1),\ \neg R(x,2),\ldots$

Теорема 20. [***Основне особине р. н. скупова***] *(1) Ако су $A, B \subseteq N^k$ полуодлучиви скупови, онда су полуодлучиви и скупови $A \cup B$ и $A \cap B$.*
(2) Ако је $A \subseteq \mathbb{N}^{k+m}$ полуодлучив скуп, онда је полуодлучив и скуп $B \subseteq \mathbb{N}^k$ дефинисан са $B = \{\vec{x} \mid \exists y_1 \ldots \exists y_m A(\vec{x}, y_1, \ldots, y_m)\}$.

Доказ. (1) Ако су $A, B \subseteq N^k$ полуодлучиви скупови, онда постоје одлучиви скупови $P, Q \subseteq \mathbb{N}^{k+1}$ такви да је $A(\vec{x}) \Leftrightarrow \exists y P(\vec{x}, y)$ и $B(\vec{x}) \Leftrightarrow \exists y Q(\vec{x}, y)$. Тада:

$$\begin{aligned} \vec{x} \in A \cup B &\Leftrightarrow \exists y P(\vec{x}, y) \vee \exists y Q(\vec{x}, y) \\ &\Leftrightarrow \exists y (P(\vec{x}, y) \vee Q(\vec{x}, y)), \end{aligned} \qquad \begin{aligned} \vec{x} \in A \cap B &\Leftrightarrow \exists y P(\vec{x}, y) \wedge \exists y Q(\vec{x}, y) \\ &\Leftrightarrow \exists y_1 \exists y_2 (P(\vec{x}, y_1) \wedge Q(\vec{x}, y_2)) \\ &\Leftrightarrow \exists y (P(\vec{x}, \langle y \rangle_1) \wedge Q(\vec{x}, \langle y \rangle_2)). \end{aligned}$$

У оба случаја жељени закључак изводимо из претходне теореме.

(2) Ако је $A \subseteq \mathbb{N}^{k+m}$ полуодлучив, онда постоји рекурзиван скуп $R \subseteq \mathbb{N}^{k+m+1}$ такав да је $A(\vec{x}, y_1, \ldots, y_m) \Leftrightarrow \exists z R(\vec{x}, y_1, \ldots, y_m, z)$. Нека је $\gamma : \mathbb{N}^{m+1} \overset{1-1}{\underset{\text{на}}{\to}} \mathbb{N}$ било које рекурзивно кодирање, чије су декодирајуће функције $\gamma_i : \mathbb{N} \to \mathbb{N}$, $1 \leq i \leq m+1$. Тада је

$$\begin{aligned} B(\vec{x}) &\Leftrightarrow \exists y_1 \ldots \exists y_m A(\vec{x}, y_1, \ldots, y_m) \\ &\Leftrightarrow \exists y_1 \ldots \exists y_m \exists z R(\vec{x}, y_1, \ldots, y_m, z) \\ &\Leftrightarrow \exists u R(\vec{x}, \gamma_1(u), \ldots, \gamma_{m+1}(u)), \end{aligned}$$

одакле следи да је B рекурзивно набројив скуп. □

Комплемент полуодлучивог скупа не мора бити полуодлучив. То ћемо једноставно показати ослањајући се на наредну теорему.

Теорема 21. *Скуп $A \subseteq \mathbb{N}^k$ је рекурзиван акко су A и $A^{\complement} = \mathbb{N}^k \setminus A$ полуодлучиви.*

Доказ. Ако је A рекурзиван, јасно је да су и A и $A^{\complement}$ рекурзивно набројиви скупови.

Докажимо и обрнуто. Ако су A и $A^{\complement}$ полуодлучиви, постоје рекурзивни скупови $P, Q \subseteq \mathbb{N}^{k+1}$ такви да је $A(\vec{x}) \Leftrightarrow \exists y P(\vec{x}, y)$ и $A^{\complement}(\vec{x}) \Leftrightarrow \exists y Q(\vec{x}, y)$. Функција

$$f(\vec{x}) = \mu y(P(\vec{x}, y) \vee Q(\vec{x}, y))$$

је парцијално рекурзивна и тотална, јер за свако $\vec{x} \in \mathbb{N}^k$ важи $\vec{x} \in A$ или $\vec{x} \in A^{\complement}$, па постоји y такав да је или $P(\vec{x}, y)$ или $Q(\vec{x}, y)$. Дакле, f је рекурзивна функција, одакле следи и да је A рекурзиван скуп будући да важи: $A(\vec{x}) \Leftrightarrow P(\vec{x}, f(\vec{x}))$. □

Последица 7. *Скуп $K^{\complement}$ није рекурзивно набројив.*

Доказ. Доказали смо да је K рекурзивно набројив скуп који није рекурзиван. Према претходној теореми $K^{\complement}$ не може бити рекурзивно набројив. □

Наредна теорема показује да се многа разматрања о парцијално рекурзивним функцијама могу свести на проучавање њихових домена, одн. рекурзивно набројивих скупова.

Теорема 22. [***Теорема о графику***][144] *Парцијална k-арна функција f је парцијално рекурзивна акко је њен график*

$$\Gamma_f = \{(\vec{x}, y) \in \mathbb{N}^{k+1} \mid f(\vec{x}) = y\}$$

рекурзиво набројив скуп.

[144] Већ смо доказали (страна 64) да је тотална функција рекурзивна акко је њен график парцијално рекурзиван.

Доказ. ($\Rightarrow$) Ако је f парцијално рекурзивна k-арна функција, онда је $f = \varphi_e^{(k)}$ за неко $e \in \mathbb{N}$. На основу Клинијеве теореме о нормалној форми за све $(\vec{x}, y) \in \mathbb{N}^{k+1}$ важи:

$$\begin{aligned}(\vec{x}, y) \in \Gamma_f &\Leftrightarrow \varphi_e^{(k)}(\vec{x}) = y \\ &\Leftrightarrow U(\mu z T_k(e, \vec{x}, z)) = y \\ &\Leftrightarrow \exists z(T_k(e, \vec{x}, z) \wedge (\forall t < z)\neg T_k(e, \vec{x}, t) \wedge U(z) = y)\end{aligned}$$

па је Γ_f рекурзивно набројив, према теореми 19.
($\Leftarrow$) Ако је Γ_f рекурзивно набројив, онда

$$(\vec{x}, y) \in \Gamma_f \Leftrightarrow \exists z R(\vec{x}, y, z),$$

за неки рекурзиван скуп $R \subseteq \mathbb{N}^{k+2}$. Дакле,

$$f(\vec{x}) \downarrow \Leftrightarrow \exists y \exists z R(\vec{x}, y, z).$$

Користећи кодирање парова природних бројева[145] добијамо

[145] $(z, y) \mapsto t = \langle z, y \rangle$

$$f(\vec{x}) \simeq \langle \mu t R(\vec{x}, \langle t \rangle_1, \langle t \rangle_2) \rangle_1,$$

одакле следи да је f парцијално рекурзивна функција. □

Пример 45. У примеру 35, на страни 64, показали смо да график парцијално рекурзивне функције, која није тотална, може бити рекурзиван скуп. Парцијална карактеристична функција скупа K,

$$\overline{\chi}_K(x) = \begin{cases} 1, & x \in K, \\ \uparrow, & x \notin K, \end{cases}$$

јесте парцијално рекурзивна, а њен график $\Gamma_{\overline{\chi}_K} = \{(x, 1) \mid x \in K\}$ није рекурзиван зато што скуп K није рекурзиван.

Одељак завршавамо тврђењем које оправдава зашто полуодлучиве скупове називамо и *рекурзивно набројивим* скуповима.

Теорема 23. *За сваки скуп $A \subseteq \mathbb{N}$ следећи услови су еквивалентни:*

1. *A је рекурзивно набројив скуп.*
2. *$A = \mathrm{ran}(\varphi_e)$, за неко $e \in \mathbb{N}$.*
3. *$A = \emptyset$ или је $A = f[\mathbb{N}]$, за неку примитивно рекурзивну функцију $f : \mathbb{N} \to \mathbb{N}$.*
4. *$A = \emptyset$ или је $A = f[\mathbb{N}]$, за неку рекурзивну функцију $f : \mathbb{N} \to \mathbb{N}$.*

Доказ. (1) $\Rightarrow$ (2) Нека је $A = \mathrm{dom}(\varphi_{e_0})$, за неко $e_0 \in \mathbb{N}$. Унарна функција f дефинисана са

$$f(x) \simeq x + \mathbf{0}(\varphi_{e_0}(x)),$$

јесте парцијално рекурзивна, па постоји $e_1 \in \mathbb{N}$ такав да је $f = \varphi_{e_1}$. Очигледно је $A = \mathrm{dom}(\varphi_{e_0}) = \mathrm{ran}(f) = \mathrm{ran}(\varphi_{e_1})$:

$$x \in A \;\Leftrightarrow\; \varphi_{e_0}(x)\downarrow \Leftrightarrow\; \varphi_{e_1}(x)\downarrow x \;\Leftrightarrow\; x \in \mathrm{ran}(\varphi_{e_1}).$$

(2) $\Rightarrow$ (3) Нека је $A = \mathrm{ran}(\varphi_{e_0})$, за неко $e_0 \in \mathbb{N}$. Претпоставимо да је $A \neq \emptyset$ и a_0 произвољно изабран елемент из A. Ослањајући се на Клинијеву теорему о нормалној форми, дефинишимо функцију $F : \mathbb{N}^2 \to \mathbb{N}$ на следећи начин

$$F(x,t) = \begin{cases} U(\mu z \leqslant t\, T_1(e_0, x, z)), & \text{ако } \exists z \leqslant t\, T_1(e_0, x, t), \\ a_0, & \text{иначе.} \end{cases}$$

Функција F је примитивно рекурзивна и важи $F[\mathbb{N}^2] = A$. Инклузија $F[\mathbb{N}^2] \subseteq A$ је очигледна. Такође, ако $a \in A$, тада постоји $x \in \mathbb{N}$ да је $\varphi_{e_0}(x) = a$, што значи да постоји и $t \in \mathbb{N}$ тако да важи $T_1(e_0, x, t)$ и

$$\varphi_{e_0}(x) = U(\mu z \leqslant t\, T_1(e_0, x, z)).$$

Дакле, $a = F(x,t) \in F[\mathbb{N}^2]$.

Тражену унарну рекурзивну функцију f дефинишемо користећи неко кодирање скупа $\mathbb{N}^2$; на пример $f(x) \stackrel{\text{def}}{=} F(\langle x\rangle_1, \langle x\rangle_2)$. Очигледно је $A = F[\mathbb{N}^2] = f[\mathbb{N}]$.
$(3) \Rightarrow (4)$ Тривијално.
$(4) \Rightarrow (1)$ Празан скуп је рекурзивно набројив (штавише, рекурзиван је).

Нека је $A = f[\mathbb{N}]$ за неку рекурзивну функцију $f : \mathbb{N} \to \mathbb{N}$ (наравно, $A \neq \emptyset$). Будући да за свако $x \in \mathbb{N}$ важи:

$$x \in A = f[\mathbb{N}] \Leftrightarrow \exists t\, f(t) = x,$$

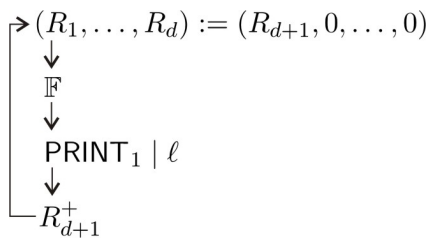
A је домен парцијално рекурзивне функције $x \mapsto \mu t\,(f(t) = x)$, одакле следи да је A рекурзивно набројив. □

Посебно је корисна еквиваленција $(1) \Leftrightarrow (4)$ претходне теореме. Непразни рекурзивно набројиви скупови су они подскупови од $\mathbb{N}$ чији се елементи могу ефективно набрајати, односно скупови за које постоји рекурзивна функција f таква да је

$$A = \{f(0), f(1), f(2), \ldots\} = \{f(n) \mid n \in \mathbb{N}\} = f[\mathbb{N}].$$

ПРИМЕР 46. Еквиваленцију $(1) \Leftrightarrow (4)$ можемо описати користећи *регистар машине са штампачем*. Ове машине, поред инструкција облика $R_k^{+1} \mid \ell$ и $R_k^{-} \mid \ell_0, \ell_1$, извршавају и инструкције облика $\mathsf{PRINT}_k \mid \ell$: штампа се садржај регистра R_k и затим се прелази на извршавање инструкције чији је редни број ℓ. Коначан низ инструкција наведеног облика називаћемо RM^+-програмом. За било који RM^+-програм $\mathbb{P}$ нека је $E_{\mathbb{P}}$ скуп свих бројева који ће бити одштампани током извршавања програма $\mathbb{P}$ када су на почетку у свим регистрима уписане нуле.

Докажимо да је скуп $A \subseteq \mathbb{N}$ рекурзивно нарбојив акко постоји RM^+-програм $\mathbb{P}$ такав да је $A = E_{\mathrm{P}}$.
$(\Rightarrow)$ Ако је $A = \emptyset$ тврђење тривијално важи. Претпоставимо да је $A = f[\mathbb{N}]$, где је f нека унарна рекурзивна функција. Нека је $\mathbb{F}$ (обичан) RM-програм који израчунава функцију f. Ако је $\|\mathbb{F}\| = d$, тражени RM^+-програм $\mathbb{P}$ конструишемо према шеми приказаној са десне стране.



$(\Leftarrow)$ Нека је $\mathbb{P}$ произвољан RM^+-програм. Покажимо да је $E_{\mathbb{P}}$ рекурзивно набројив. Довољно је написати RM-програм $\mathbb{E}$ који ће се заустављати само за бројеве који припадају $E_{\mathbb{P}}$. Нека је $\|\mathbb{P}\| = d$. Програм $\mathbb{E}$ конструишемо на следећи начин: најпре улаз x (унет у R_1) снимимо у R_{d+1} ($R_{d+1} := R_1$), затим обришемо регистар R_1 и покренемо програм $\mathbb{P}'$ добијен из P тако што је свака инструкција облика $\mathsf{PRINT}_k \mid \ell$ замењена (пот)програмом којим се упоређују садржаји регистара R_k и R_{d+1} и

- ако су једнаки, онда се $\mathbb{E}$ зауставља и даје резулатат 1;
- ако су различити, онда се прелази на инструкцију која одговара ℓ-тој инструкцији програма $\mathbb{P}$.

Очигледно, $\mathbb{E}$ за задато x израчунава $\overline{\chi}_{E_{\mathbb{P}}}(x)$, тако што памти x и упоређује га са сваким бројем који штампа $\mathbb{P}$: ако број x буде икада одштампан, $\mathbb{E}$ ће се зауставити и вратити резулатат 1, а у супротном се неће зауставити.

Пример 47. Покажимо да скуп $\text{Tot} = \{e \mid \text{dom}(\varphi_e) = \mathbb{N}\}$ није рекурзивно набројив.

Претпоставимо супротно: Tot је рекурзивно набројив. Како је $\text{Tot} \neq \emptyset$, постоји рекурзивна функција f таква да је $\text{Tot} = f[\mathbb{N}]$ (f је функција ефективно набраја елементе скупа Tot). Дефинишимо функцију

$$g(x) \simeq \Phi_U(f(x), x) + 1.$$

Јасно, g је парцијално рекурзивна функција. Штавише, g је и тотална, јер за свако $x \in \mathbb{N}$, $\Phi_U(f(x), x) \downarrow$, будући да је $\varphi_{f(x)}$ тотална функција. Дакле, g је рекурзивна функција, па пошто f набраја индексе свих рекурзивних функција, онда постоји k такав да је $g = \varphi_{f(k)}$. Из последње једнакости специјално следи да је $g(k) = \varphi_{f(k)}(k)$, што није могуће, јер према дефиницији функције g важи $g(k) = \Phi_U(f(x), x) + 1 = \varphi_{f(k)}(k) + 1$. Контрадикција.

Дакле, није могуће ефективно набрајати индексе свих рекурзивних функција (тј. програме који се заустављају за сваки улаз).

Теорема 23 даје неке еквивалентне карактеризације рекурзивно набројивих подскупова од $\mathbb{N}$. Користећи неко примитивно рекурзивно кодирање $\gamma : \mathbb{N}^k \overset{1-1}{\underset{\text{на}}{\rightarrow}} \mathbb{N}$, чије су декодирајуће функције $\gamma_i : \mathbb{N} \to \mathbb{N}$ (такође примитивно рекурзивне), једноставно се доказује да за сваки непразан рекурзивно набројив подскуп $A \subseteq \mathbb{N}^k$, постоји рекурзивна функција $f : \mathbb{N} \to \mathbb{N}$ која набраја γ-кодове елемената из A: $\vec{x} \in A$ акко постоји k такво да је $f(k) = \gamma(\vec{x})$. Ово директно следи из претходне теореме и чињенице: A је рекурзивно набројив акко је скуп $\overline{A} = \{\gamma(\vec{x}) \mid \vec{x} \in A\}$ рекурзивно набројив.

Аритметичка хијерархија

Дефиниција 13. *Скуп $A \subseteq \mathbb{N}^k$ је аритметички ако је постоји рекурзиван скуп $R \subseteq \mathbb{N}^{k+m}$ такав да за све $\vec{x} \in \mathbb{N}^k$ важи*

$$(*) \qquad A(\vec{x}) \Leftrightarrow \mathrm{Q}_1 y_1 \cdots \mathrm{Q}_m y_m R(\vec{x}, y_1, \ldots, y_m),$$

где је $\mathrm{Q}_i y_i$ или $\exists y_i$ или $\forall y_i$, тј. $\mathrm{Q}_i \in \{\exists, \forall\}$, $1 \leqslant i \leqslant m$.

Формулу $(*)$ из претходне дефиниције називамо *експлицитном дефиницијом* скупа A. У формули са леве стране еквиваленције $(*)$ део $\mathrm{Q}_1 y_1 \cdots \mathrm{Q}_m y_m$ се назива *префикс*, а $R(\vec{x}, \vec{y})$ *матрикс* одговарајуће формуле. Пре свега смо заинтересовани за префикс, јер он мери колико је скуп A „далеко“ од рекурзивног скупа. Приметимо најпре да се узастопна појављивања истих квантификатора у префиксу могу једноставно елиминисати:

$$\exists x \exists y P(x,y) \Leftrightarrow \exists z P(\langle z \rangle_1, \langle z \rangle_2) \quad \text{и} \quad \forall x \forall y P(x,y) \Leftrightarrow \forall z P(\langle z \rangle_1, \langle z \rangle_2).$$

Овакву врсту поједностављивања префикса називаћемо *контракцијом* квантификатора.

Префикс је **алтернирајући** ако се у њему не појављују један до другог два иста квантификатора. Префикс је Π_n ако је алтернирајући, садржи n квантификатора и први (слева) квантификатор је $\forall$. Префикс је Σ_n ако је алтернирајући, садржи n квантификатора и први (слева) квантификатор је $\exists$.

Π_1: $\forall$; Π_2: $\forall\exists$; Π_3: $\forall\exists\forall$; ...
Σ_1: $\exists$; Σ_2: $\exists\forall$; Σ_3: $\exists\forall\exists$; ...

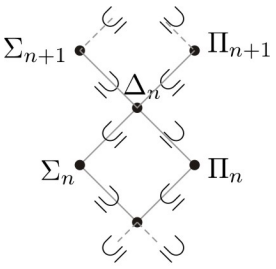
Дефиниција 14. *Скуп (релација) A је Π_n ако се може експлицитно дефинисати формулом чији је префикс Π_n, а матрикс је рекурзиван.*

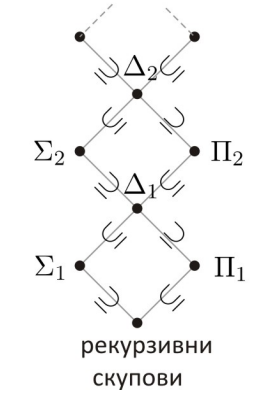
Скуп (релација) A је Σ_n ако се може експлицитно дефинисати формулом чији је префикс Σ_n, а матрикс је рекурзиван.

Скуп (релација) A је Δ_n ако је и Π_{n+1} и Σ_{n+1}.

Ако ознаке Π_n, Σ_n и Δ_n искористимо да означимо редом скупове свих Π_n, Σ_n и Δ_n релација, онда је $\Delta_n = \Pi_{n+1} \cap \Sigma_{n+1}$.

Класификација аритметичких релација помоћу Π_n и Σ_n префикса се назива **аритметичка хијерархија**. Увођењем тзв. *лажних* променљивих, тј. променљивих које се заједно са неким квантификатором појављују у префиксу, али их нема у матриксу, једноставно закључујемо да важе инклузије приказане на слици десно. У наставку ћемо показати да је хијерархија „права“, што значи да су све инклузије заправо строге.





Нека је $\mathcal{R}$ неки скуп k-арних релација. Кажемо да $(k+1)$-арна релација E_k *универзална* за релације из $\mathcal{R}$ уколико за све $R \in \mathcal{R}$ постоји $e \in \mathbb{N}$ такав да за све $\vec{x} \in \mathbb{N}^k$ важи $R(\vec{x}) \Leftrightarrow E_k(\vec{x}, e)$.

Теорема 24. *За све $n, k \geqslant 1$, постоји $(k+1)$-арна Σ_n релација S_k^n која је универзална за скуп свих k-арних Σ_n релација.*

За све $n, k \geqslant 1$, постоји $(k+1)$-арна Π_n релација P_k^n која је универзална за скуп свих k-арних Π_n релација.

Доказ. Математичком индукцијом по n доказујемо да за све $k \geqslant 1$ постоји $(k+1)$-арна Σ_n релација S_k^n која је универзална за све k-арне Σ_n релације, и $(k+1)$-арна Π_n релација P_k^n која је универзална за све k-арне Π_n релације.

База индукције: $n = 1$. Ако је A произвољна k-арна Σ_1 релација, онда постоји рекурзивана релација $R \subseteq \mathbb{N}^{k+1}$ таква да за све $\vec{x} \in \mathbb{N}^k$ важи $A(\vec{x}) \Leftrightarrow \exists u R(\vec{x}, u)$. Ако је e_0 индекс функције χ_R, онда према Клинијевој теореми о нормалној форми важи:

$$R(\vec{x}, u) \Leftrightarrow \exists z \left(T_{k+1}(e_0, \vec{x}, u, z) \wedge U(z) = 1\right).$$

Дакле, за све $\vec{x} \in \mathbb{N}^k$ важи $A(\vec{x}) \Leftrightarrow \exists u \exists z \left(T_{k+1}(e_0, \vec{x}, u, z) \wedge U(z) = 1\right)$. Последња еквиваленција указује на то како треба дефинисати $(k+1)$-арну Σ_1 релацију S_k^1 која је универзална за све k-арне Σ_1 релације:

$$S_k^1(\vec{x}, e) \overset{\text{def}}{\Leftrightarrow} \exists y \left(T_{k+1}(e, \vec{x}, \langle y \rangle_1, \langle y \rangle_2) \wedge U(\langle y \rangle_2) = 1\right).$$

Једноставно се уочава да $(k+1)$-арну Π_1 релацију P_k^1, која је универзална за све k-арне Π_1 релације, можемо дефинисати на следећи начин:

$$P_k^1(\vec{x}, e) \overset{\text{def}}{\Leftrightarrow} \neg S_k^1(\vec{x}, e).$$

Индуктивна претпоставка: за све $k \geqslant 1$ постоји $(k+1)$-арна Σ_n релација S_k^n која је универзална за све k-арне Σ_n релације, и $(k+1)$-арна Π_n релација P_k^n која је универзална за све k-арне Π_n релација.

Нека је A произвољна k-арна Σ_{n+1} релација. Тада постоји $(k+1)$-арна Π_n релација R таква да за све $\vec{x} \in \mathbb{N}^k$ важи: $A(\vec{x}) \Leftrightarrow \exists u R(\vec{x}, u)$. Према индуктивној претпоставци постоји $(k+2)$-арна Π_n релација P_{k+1}^n која је универзална за све $(k+1)$-арне Π_n релације. То значи да постоји e_0 такво да $R(\vec{x}, u) \Leftrightarrow P_{k+1}^n(\vec{x}, u, e_0)$. Дакле, за све $\vec{x} \in \mathbb{N}^k$ важи

$$A(\vec{x}) \Leftrightarrow \exists u P_{k+1}^n(\vec{x}, u, e_0).$$

Одавде закључујемо да $(k+1)$-арну Σ_{n+1} релацију S_k^{n+1}, која је универзална за све k-арне Σ_{n+1} релације, можемо дефинисати на следећи начин:

$$S_k^{n+1}(\vec{x}, e) \overset{\text{def}}{\Leftrightarrow} \exists u P_{k+1}^n(\vec{x}, u, e).$$

Тражена $(k+1)$-арна Π_{n+1} релација P_k^{n+1}, универзална за све k-арне Π_{n+1} релације, може се дефинисати еквиваленцијом:

$$P_k^{n+1}(\vec{x}, e) \overset{\text{def}}{\Leftrightarrow} \neg S_{k+1}^n(\vec{x}, e).$$

□

Теорема 25. [***Основна теорема о аритметичкој хијерархији***] *За свако* $n \geqslant 1$, *постоји унарна* Π_n *релација која није* Σ_n, *па самим тим није* Π_k *нити* Σ_k, *за било које* $k < n$. *Такође, за свако* $n \geqslant 1$, *постоји унарна* Σ_n *релација која није* Π_n, *па самим тим није* Π_k *нити* Σ_k, *за било које* $k < n$.

Доказ. Нека је P_1^n бинарна Π_n релација која набраја све унарне Π_n релације. Дефинишимо унарну релацију D:

$$D(x) \overset{\text{def}}{\Leftrightarrow} P_1^n(x, x), \ x \in \mathbb{N}.$$

Није тешко закључити да је D заправо Π_n релација, али да $D^{\complement}$ није Π_n, одакле следи да D не може бити Σ_n. Аналогно се доказује други део теореме. □

Једна од последица претходне теореме јесте да тврђење аналогно теореми 24 не важи за Δ_n скупове: не постоји бинарна Δ_n релација која набраја скуп свих унарних Δ_n релација.

Теорема параметризације

Ефективно кодирање програма омогућава да се кодови RM-програма појављују као улази других RM-програма, а тиме и аритметизацију неких општих поступака који се односе на програме. На пример, поступак надовезивања два RM-програма описали смо на страни 29. Реч је о општем поступку којим се за два дата програма $\mathbb{P}_1$ и $\mathbb{P}_2$ саставља нови програм $\mathbb{P}_1\mathbb{P}_2$. Интуитивно је јасно (између осталог и према Черч-Тјуринговој тези) да се поступак може аритметизовати једном примитивно рекурзивном функцијом $*_{\text{RM}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ таквом да је $[\![\mathbb{P}_1]\!] *_{\text{RM}} [\![\mathbb{P}_2]\!] = [\![\mathbb{P}_1\mathbb{P}_2]\!]$.

Лема 11. *Постоји примитивно рекурзивна бинарна функција* $*_{\text{RM}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *таква да за свака два* RM*-програма* $\mathbb{P}_1$ *и* $\mathbb{P}_2$ *важи једнакост:* $[\![\mathbb{P}_1]\!] *_{\text{RM}} [\![\mathbb{P}_2]\!] = [\![\mathbb{P}_1\mathbb{P}_2]\!]$.

Ову лему нећемо строго доказивати, јер се доказ своди на састављање компликованог израза који дефинише поменуту функцију и потврђује њену примитивну рекурзивност, а сам по себи није важан за даљу причу.[150]

Уопште, сваку (примитивно) рекурзивну функцију $f : \mathbb{N}^k \to \mathbb{N}$ можемо посматрати као аритметичку варијанту опште процедуре састављања RM-програма у зависности од вредности неких k параметара. Можемо замислити да за сваку k-торку $(n_1, \ldots, n_k)$, вредност $f(n_1, \ldots, n_k)$ представља заправо кôд одговарајућег RM-програма, што значи да функција f у извесном смислу описује поступак састављања одређене врсте RM-програма.

Пример 48. Посматрајмо следеће РМ-програме:

$$\mathbb{C}_0 \left\{ \begin{array}{lll} 1. & R_1^- \mid 2,1; \end{array} \right. \qquad \mathbb{C}_n \left\{ \begin{array}{lll} 1. & R_1^- \mid 2,1 & \\ 2. & R_1^+ \mid 3 & \\ & \vdots & n = 1,2,3,4,\ldots \\ n+1. & R_1^+ \mid n+2 & \end{array} \right.$$

Очигледно, $\mathbb{C}_n$ израчунава константну функцију $x \mapsto n$. Најважније је приметити да смо низ програма $(\mathbb{C}_n)$ униформно описали, тј. навели општи поступак како се за задато n саставља програм који израчунава константну функцију $x \mapsto n$. Користећи уведено кодирање програма (страна 66) није тешко показати да је $n \mapsto [\![\mathbb{C}_n]\!]$ примитивно рекурзивна функција.[151] Ову функцију можемо схватити и као ефективно генерисање (бирање) по једног индекса за сваку константу функцију. (Подсећамо да свака парцијално рекурзивна функција има бесконачно много индекса.)

Из претходних разматрања издвајамо следећи закључак: постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $x, y \in \mathbb{N}$ важи

[150] Неформално, вредност $x *_{\text{RM}} y$ рачунамо тако што прво нађемо програме $\mathbb{P}_1 = [\![x]\!]^{-1}$ и $\mathbb{P}_2 = [\![y]\!]^{-1}$, затим их надовежемо (строго се придржавајући правила надовезивања са стране 29) и најзад одредимо $[\![\mathbb{P}_1\mathbb{P}_2]\!]$. Очигледно је да током израчунавања вредности $x *_{\text{RM}} y$ нема потребе за неограниченом минимизацијом, па закључујемо да је $*_{\text{RM}}$ примитивно рекурзивна.

[151] Како је $[R_1^- \mid 2,1] = \langle 1, \langle 2,1\rangle\rangle = 45$ и $[R_1^+ \mid \ell] = \langle 0, \ell\rangle = 2\ell$, имамо да је:

$$\begin{aligned} [\![\mathbb{C}_n]\!] &= \lceil 45, 6, 8, \ldots, 2(n+2)\rceil \\ &= 2^{45} + 2^{45+6+1} + \cdots + 2^{45+6+\cdots+2(n+2)+n} \\ &= 2^{45} + \sum_{i=1}^{n} 2^{45+i+\sum_{j=1}^{i} 2(j+2)} \end{aligned}$$

$\varphi_{h(x)}(y) = x$. У овом примеру смо и дефинисали једну такву функцију: $h(x) \stackrel{\text{def}}{=} [\![\mathbb{C}_x]\!]$; за све $x, y \in \mathbb{N}$, $\varphi_{[\![\mathbb{C}_x]\!]}(y) = \Phi_U([\![\mathbb{C}_x]\!], y) = x$.

ПРИМЕР 49. Да ли постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $n, x \in \mathbb{N}$ важи $\varphi_{h(n)}(x) \simeq n^x$? Другим речима, да ли се ефективно могу генерисати индекси експоненцијалних функција:

$$x \mapsto 0^x,\ x \mapsto 1^x,\ x \mapsto 2^x,\ x \mapsto 3^x,\ \cdots$$

Свака од функција наведеног низа добија се фиксирањем прве координате примитивно рекурзивне функције $\exp : \mathbb{N}^2 \to \mathbb{N}$, $\exp(n, x) = n^x$. Нека је $\mathbb{E}$ програм који је израчунава. Када на програм[152] $\mathbb{P}_n$, који садржај регистра R_1 копира у R_2 а затим у R_1 уписује број n, надовежемо $\mathbb{E}$, добијамо програм $\mathbb{E}_n = \mathbb{P}_n\mathbb{E}$ који израчунава унарну функцију $x \mapsto n^x$. Није тешко закључити да је функција $n \mapsto [\![\mathbb{E}_n]\!] = [\![\mathbb{P}_n]\!] *_{\text{RM}} [\![\mathbb{E}]\!]$ примитивно рекурзивна, као и да је то тражена функција: $h(n) \stackrel{\text{def}}{=} [\![\mathbb{E}_n]\!]$. Заиста, за све $n, x \in \mathbb{N}$ важи:

$$\varphi_{h(n)}(x) = \varphi_{[\![\mathbb{E}_n]\!]}(x) = \Phi_U([\![\mathbb{E}_n]\!], x) = n^x.$$

[152] $\mathbb{P}_n$
$$\begin{cases} 1. & R_1^- \mid 3,2 \\ 2. & R_2^+ \mid 1 \\ 3. & R_1^+ \mid 4 \\ 4. & R_1^+ \mid 5 \\ & \vdots \\ n+2. & R_1^+ \mid n+3 \end{cases}$$

У наставку уопштавамо запажања из претходних примера. Нека су $m, n \geqslant 1$ фиксирани природни бројеви. За сваки RM-програм $\mathbb{P}$ и $\vec{x} \in \mathbb{N}^m$ нека је $\mathbb{S}_n^m(\mathbb{P}, \vec{x})$ програм добијен надовезивањем редом програма:

$$\begin{cases} 1.\ R_n^- \mid 3,2 \\ 2.\ R_{m+n}^+ \mid 1 \end{cases} \cdots \begin{cases} 1.\ R_1^- \mid 3,2 \\ 2.\ R_{m+1}^+ \mid 1 \end{cases} \begin{cases} 1.\ R_1^+ \mid 2 \\ \vdots \\ x_1.\ R_1^+ \mid x_1 + 1 \end{cases} \cdots \begin{cases} 1.\ R_m^+ \mid 2 \\ \vdots \\ x_m.\ R_m^+ \mid x_m + 1 \end{cases} \mathbb{P}$$

Израчунавање програма $\mathbb{S}_n^m(\mathbb{P}, \vec{x})$ за почетну конфигурацију облика $(y_1, \ldots, y_n)$ може се описати на следећи начин:

$$(y_1, \ldots, y_n) \to \cdots \to (\underbrace{0, \ldots, 0}_{m \text{ нула}}, y_1, \ldots, y_n) \to \cdots \to (x_1, \ldots, x_m, y_1, \ldots, y_n) \to_{\mathbb{P}} \cdots$$

Одавде закључујемо да $\mathbb{S}_n^m(\mathbb{P}, \vec{x})(\vec{y}) \downarrow$ акко $\mathbb{P}(\vec{x}, \vec{y}) \downarrow$; штавише:

$$\varphi^{(n)}_{\mathbb{S}_n^m(\mathbb{P}, \vec{x})}(\vec{y}) \simeq \varphi^{(m+n)}_{\mathbb{P}}(\vec{x}, \vec{y}).$$

Наведени поступак састављања програма $\mathbb{S}_n^m(\mathbb{P}, \vec{x})$ за дати програм $\mathbb{P}$ и дати вектор $\vec{x} \in \mathbb{N}^m$, оправдава уверење да је функција $s_n^m : \mathbb{N}^{m+1} \to \mathbb{N}, s_n^m(e, \vec{x}) = [\![\mathbb{S}_n^m([\![e]\!]^{-1}, \vec{x})]\!]$ примитивно рекурзивна.[153] Без улажења у формалне детаље, у наредној теореми сумирамо претходне закључке.

Теорема 26. [***Теорема параметризације***[154]] *За било која два природна броја $m, n \geqslant 1$, постоји примитивно рекурзивна функција $s_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ таква да за све $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$, $\vec{y} \in \mathbb{N}^n$ важи једнакост:*

$$\varphi^{(n)}_{s_n^m(e, \vec{x})}(\vec{y}) \simeq \varphi^{(m+n)}_{e}(\vec{x}, \vec{y}).$$

[153] Неформално, вредност $s_n^m(e, \vec{x})$ рачунамо тако што: прво нађемо RM-програм $\mathbb{P} = [\![e]\!]^{-1}$, затим напишемо програм $\mathbb{S}_n^m(\mathbb{P}, \vec{x})$ (као што је описано) и најзад одредимо кôд овог програма. Очигледно је да током израчунавања вредности функције s_n^m нема потребе за неограниченом минимизацијом, па закључујемо да је s_n^m примитивно рекурзивна.

[154] Теорема параметризације се назива и s-m-n-теорема.

Осврнимо се на формулацију претходне теореме у једном специјалном случају. Нека је f било која бинарна парцијална функција и нека је e њен индекс: $f(x,y) \simeq \varphi_e^{(2)}(x,y)$, за све $x,y \in \mathbb{N}$. За сваку фиксирану вредност првог аргумента, природно је дефинисана по једна унарна, такође парцијално рекурзивна, функција:

$$y \mapsto f(0,y),\ y \mapsto f(1,y),\ y \mapsto f(2,y),\ y \mapsto f(3,y),\ y \mapsto f(4,y),\ \cdots$$

Свака од функција наведеног низа има своје индексе. Теорема параметризације, у случају $m = n = 1$, тврди да постоји примитивно рекурзивна функција $s_1^1 : \mathbb{N}^2 \to \mathbb{N}$ која одређује (бира) по један индекс за сваку функција наведеног низа и то на основу индекса e функције f и фиксиране вредности првог аргумента x:

функција:	$y \mapsto f(0,y)$	$y \mapsto f(1,y)$	$y \mapsto f(2,y)$	$\cdots$
индекс:	$s_1^1(e,0)$	$s_1^1(e,1)$	$s_1^1(e,2)$	$\cdots$

Да бисмо што јасније истакли значај и применљивост Теореме параметризације, главне закључке Примера 48 и Примера 49 изводимо применом поменуте теореме.

Пример 50. Докажимо да постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $x,y \in \mathbb{N}$ важи $\varphi_{h(x)}(y) \simeq x$.

Тражи се примитивно рекурзивна функција h која генерише индексе константних функција: $x \mapsto 0,\ x \mapsto 1,\ x \mapsto 2,\ x \mapsto 3,\ \cdots$ Пројекција $\Pi_1^2 : \mathbb{N}^2 \to \mathbb{N}$, $\Pi_1^2(x,y) = x$, јесте примитивно рекурзивна, па има неки индекс e_0: $\Pi_1^2 = \varphi_{e_0}^{(2)}$. Нека је $h(x) \overset{\text{def}}{=} s_1^1(e_0,x)$, где је s_1^1 примитивно рекурзивна функција чије постојање тврди теорема параметризације за $m = n = 1$. Тада за свако $x \in \mathbb{N}$

$$\varphi_{h(x)}(y) \simeq \varphi_{s_1^1(e_0,x)}(y) \simeq \varphi_{e_0}^{(2)}(x,y) \simeq \Pi_1^2(x,y) = x.$$

Дакле, поново смо доказали оно што се тврди на крају Примера 48.

Пример 51. Докажимо да постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $n,x \in \mathbb{N}$ важи $\varphi_{h(n)}(x) \simeq n^x$.

Очигледно, тражи се примитивно рекурзивна функција h која генерише индексе експоненцијалних функција: $x \mapsto 0^x,\ x \mapsto 1^x,\ x \mapsto 2^x,\ x \mapsto 3^x,\ \cdots$ Свака од функција наведеног низа добија се фиксирањем прве координате функције $\exp : \mathbb{N}^2 \to \mathbb{N}$, $\exp(n,x) = n^x$. Функција $\exp$ је примитивно рекурзивна, па нека је e_0 њен индекс: $\exp = \varphi_{e_0}^{(2)}$. Није тешко уочити да је $h(n) \overset{\text{def}}{=} s_1^1(e_0,n)$, где је s_1^1 примитивно рекурзивна функција чије постојање тврди теорема параметризације за $m = n = 1$: за свако $x \in \mathbb{N}$,

$$\varphi_{h(n)}(x) \simeq \varphi_{s_1^1(e_0,n)}(x) \simeq \varphi_{e_0}^{(2)}(n,x) \simeq \exp(n,x) = n^x.$$

ЗАДАТАК 1. Доказати да постоји примитивно рекурзивна функција:
(а) $h : \mathbb{N} \to \mathbb{N}$, таква да за све $x, y_1, y_2 \in \mathbb{N}$ важи $\varphi^{(2)}_{h(x)}(y_1, y_2) \simeq x^2$;
(б) $h : \mathbb{N}^2 \to \mathbb{N}$, таква да за све $x_1, x_2, y \in \mathbb{N}$ важи $\varphi_{h(x_1,x_2)}(y) \simeq x_1 + x_2$;
(в) $h : \mathbb{N}^2 \to \mathbb{N}$, таква да за све $x_1, x_2, y_1, y_2 \in \mathbb{N}$ важи $\varphi^{(2)}_{h(x_1,x_2)}(y_1, y_2) \simeq \max\{x_1 y_2, x_2 y_1\}$.

ПРИМЕР 52. Докажимо да постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $x \in \mathbb{N}$ важи $\mathrm{dom}(\varphi_{h(x)}) = \{x\}$.

Функција f дата са

$$f(x,y) = \begin{cases} x & x = y, \\ \uparrow, & x \neq y, \end{cases}$$

јесте парцијално рекурзивна. Нека је e_0 индекс функције f, тј. $f = \varphi^{(2)}_{e_0}$.
Ако је $h(x) \overset{\text{def}}{=} s^1_1(e_0, x)$, онда је

$$\varphi_{h(x)}(y) \simeq \varphi_{s^1_1(e_0,x)}(y) \simeq \varphi^{(2)}_{e_0}(x,y) \simeq f(x,y).$$

Специјално, $\varphi_{h(x)}(y)\downarrow \Leftrightarrow\ f(x,y)\downarrow \Leftrightarrow\ x = y$, односно

$$y \in \mathrm{dom}(\varphi_{h(x)}) \ \Leftrightarrow\ y \in \{x\}.$$

ЗАДАТАК 2. Доказати да постоји примитивно рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $x \in \mathbb{N}$ важи $\mathrm{dom}(\varphi_{h(x)}) = \{x, x^2, x^3, x^4, \ldots\}$.

ЗАДАТАК 3. Доказати да постоји примитивно рекурзивна функција $h : \mathbb{N}^2 \to \mathbb{N}$ таква да за све $x_1, x_2 \in \mathbb{N}$ важи $\varphi^{(2)}_{h(x_1,x_2)} = \mathrm{Rec}^2(\varphi_{x_1}, \varphi^{(3)}_{x_2})$.

Теорема рекурзије

Теорему рекурзије, која спада у најзначајније теорема области којом се бавимо, мотивишемо и илуструјемо једним занимљивим примером.

Пример 53. [**Програм** $\mathbb{SELF}$] Саставићемо програм који брише било који улаз $x \in \mathbb{N}$ и као резултат враћа сопствени кôд.

Нека је $\mathbf{c} : \mathbb{N} \to \mathbb{N}$ функција $\mathbf{c}(x) = [\![\mathbb{C}_x]\!]$ из примера 48. Како су $\mathbf{c}$ и $*_{\mathrm{RM}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ примитивно рекурзивне функције[155], таква је и функција $\mathbf{c}^* : \mathbb{N} \to \mathbb{N}$ дата са $\mathbf{c}^*(x) = \mathbf{c}(x) *_{\mathrm{RM}} x$. Дакле, постоји RM-програм $\mathbb{D}$ који израчунава функцију $\mathbf{c}^*$. Нека је $\mathbb{SELF}$ програм добијен надовезивањем програма $\mathbb{C}_{[\![\mathbb{D}]\!]}$ и $\mathbb{D}$: $\mathbb{SELF} = \mathbb{C}_{[\![\mathbb{D}]\!]}\mathbb{D}$. Посматрајмо израчунавање програма $\mathbb{SELF}$ за улаз $x \in \mathbb{N}$:

$$\begin{aligned}(x,0,0,\cdots) &\to^*_{\mathbb{C}_{[\![\mathbb{D}]\!]}} ([\![\mathbb{D}]\!],0,0,\cdots)\\ &\to^*_{\mathbb{D}} (\mathbf{c}([\![\mathbb{D}]\!]) *_{\mathrm{RM}} [\![\mathbb{D}]\!],\cdots) = ([\![\mathbb{C}_{[\![\mathbb{D}]\!]}]\!] *_{\mathrm{RM}} [\![\mathbb{D}]\!],\cdots)\\ &= ([\![\mathbb{C}_{[\![\mathbb{D}]\!]}\mathbb{D}]\!],\cdots) = ([\![\mathbb{SELF}]\!],\cdots)\end{aligned}$$

Све у свему: $(x,0,0,\cdots) \to^*_{\mathbb{SELF}} ([\![\mathbb{SELF}]\!],\cdots)$.

Идеје приказане у претходном примеру могу се уопштити.

Теорема 27. [***Теорема рекурзије***[156]] *Нека је f произвољна $(k+1)$-арна парцијално рекурзивна функција. Тада постоји $e_0 \in \mathbb{N}$ такав да је $\varphi^{(k)}_{e_0}(\vec{x}) \simeq f(e_0,\vec{x})$, за свако $\vec{x} \in \mathbb{N}^k$.*

Доказ. **I начин**. Треба заправо доказати да постоји програм $\mathbb{R}$ такав да је $\varphi^{(k)}_{\mathbb{R}}(\vec{x}) \simeq f([\![\mathbb{R}]\!],\vec{x})$. Овакав програм конструишемо слично као програм $\mathbb{SELF}$. Међутим, потребно је да донекле модификујемо програме које смо користили у претходном примеру.

- За $x \in \mathbb{N}$, нека су $\mathbb{C}^{(k)}_x$ програми такви да за све $\vec{x} \in \mathbb{N}^k$ важи:

$$(x_1,\ldots,x_k,0,0,\cdots) \to^*_{\mathbb{C}^{(k)}_x} (x,x_1,\ldots,x_k,0,0,\ldots).$$

  Поред тога, низ ових програма треба да буде такав да функција $\mathbf{c}^{(k)} : \mathbb{N} \to \mathbb{N}$, $\mathbf{c}^{(k)}(x) = [\![\mathbb{C}^{(k)}_x]\!]$ буде примитивно рекурзивна.

- Нека је $\mathbb{D}^{(k)}$ програм такав да за све $x \in \mathbb{N}$ и $\vec{x} \in \mathbb{N}^k$ важи:

$$(x,x_1,\ldots,x_k,0,0,\cdots) \to^*_{\mathbb{D}^{(k)}} (\mathbf{c}^{(k)}(x) *_{\mathrm{RM}} x,x_1,\ldots,x_k,0,0,\ldots).$$

Нека је $\mathbb{F}$ програм који израчунава функцију f. Тражени програм $\mathbb{R}$ добијамо надовезивањем програма $\mathbb{C}^{(k)}_{[\![\mathbb{D}^{(k)}\mathbb{F}]\!]}$ и програма $\mathbb{D}^{(k)}\mathbb{F}$.

[155] $\mathbf{c}(x)$ јесте кôд програма који израчунава константу функцију одређену бројем x.
$*_{\mathrm{RM}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ је примитивно рекурзивна функција таква да за свака два програма $\mathbb{P}_1$ и $\mathbb{P}_2$ важи једнакост: $[\![\mathbb{P}_1]\!] *_{\mathrm{RM}} [\![\mathbb{P}_2]\!] = [\![\mathbb{P}_1\mathbb{P}_2]\!]$.

[156] Ова теорема се назива и Друга теорема рекурзије.

Посматрајмо израчунавање програма $\mathbb{R}$ за улаз $\vec{x} \in \mathbb{N}$:

$$\begin{aligned}(x_1,\ldots,x_k,0,0,\cdots) &\to^*_{\mathbb{C}^{(k)}_{[\![\mathbb{D}^{(k)}\mathbb{F}]\!]}} ([\![\mathbb{D}^{(k)}\mathbb{F}]\!],x_1,\ldots,x_k,0,0,\cdots)\\ &\to^*_{\mathbb{D}^{(k)}} (\mathbf{c}^{(k)}([\![\mathbb{D}^{(k)}\mathbb{F}]\!]) *_{\mathrm{RM}} [\![\mathbb{D}^{(k)}\mathbb{F}]\!],x_1,\ldots,x_k,0,0,\ldots)\\ &= ([\![\mathbb{C}^{(k)}_{[\![\mathbb{D}^{(k)}\mathbb{F}]\!]}]\!] *_{\mathrm{RM}} [\![\mathbb{D}^{(k)}\mathbb{F}]\!],x_1,\ldots,x_k,0,0,\ldots)\\ &= ([\![\mathbb{C}^{(k)}_{[\![\mathbb{D}^{(k)}\mathbb{F}]\!]}(\mathbb{D}^{(k)}\mathbb{F})]\!],x_1,\ldots,x_k,0,0,\ldots)\\ &= ([\![\mathbb{R}]\!],x_1,\ldots,x_k,0,0,\ldots) \to_{\mathbb{F}} \cdots\end{aligned}$$

Дакле, $\mathbb{R}$ се зауставља за улаз $(x_1,\ldots,x_k,0,0,\cdots)$ акко се $\mathbb{F}$ зауставља за $([\![\mathbb{R}]\!],x_1,\ldots,x_k,0,0,\ldots)$ и резултати ових израчунавања су исти:

$$\varphi^{(k)}_{\mathbb{R}}(\vec{x}) \simeq \varphi^{(k+1)}_{\mathbb{F}}([\![\mathbb{R}]\!],\vec{x}).$$

Очигледно, број e_0, чије постојање тврди теорема, јесте кôд програма $\mathbb{R}$.

II начин. Према теореми параметризације, постоји примитивно рекурзивна функција $s^1_k : \mathbb{N}^2 \to \mathbb{N}$ таква да за све $e \in \mathbb{N}, x \in \mathbb{N}, \vec{x} \in \mathbb{N}^k$ важи једнакост: $\varphi^{(k)}_{s^1_k(e,x)}(\vec{x}) \simeq \varphi^{(k+1)}_e(x,\vec{x})$. Парцијална $(k+1)$-арна функција g дата са $g(x,\vec{x}) \simeq f(s^1_k(x,x),\vec{x})$ јесте парцијално рекурзивна, па има неки индекс a: $g = \varphi^{(k+1)}_a$. Доказаћемо да је $e_0 = s^1_k(a,a)$ тражени природан број:

$$\begin{aligned}\varphi^{(k)}_{e_0}(\vec{x}) &\simeq \varphi^{(k)}_{s^1_k(a,a)}(\vec{x}) \simeq \varphi^{(k+1)}_a(a,\vec{x})\\ &\simeq g(a,\vec{x}) \simeq f(s^1_k(a,a),\vec{x}) \simeq f(e_0,\vec{x}).\end{aligned}$$

□

Пример 54. Применом теореме рекурзије доказаћемо да је Акерманова функција парцијално рекурзивна:

$$\left|\begin{array}{l} A(0,n) = n+1,\\ A(m+1,0) = A(m,1)\\ A(m+1,n+1) = A(m,A(m+1,n))\end{array}\right.$$

Дефинишимо једну помоћну функцију дужине три:

$$\alpha(e,m,n) \simeq \begin{cases} n+1, & m = 0,\\ \varphi^{(2)}_e(m \dot{-} 1,1), & m > 0, n = 0,\\ \varphi^{(2)}_e(m \dot{-} 1,\varphi^{(2)}_e(m,n \dot{-} 1)), & m > 0, n > 0.\end{cases}$$

Функција α је парцијално рекурзивна, па по теореми рекурзије постоји $e_0 \in \mathbb{N}$ такав да је $\alpha(e_0,m,n) \simeq \varphi^2_{e_0}(m,n)$, за све $m,n \in \mathbb{N}$. Није тешко показати да функција $\varphi^2_{e_0}$ задовољава исте једнакости којима је уведена функција A, па мора бити $A = \varphi^2_{e_0}$.

У наредном примеру илуструјемо како се применом теореме може доказати да неки скуп није рекурзиван.

ПРИМЕР 55. Скуп $H = \{(e,x) \mid \varphi_e(x)\downarrow\}$ није рекурзиван. Приметимо најпре да нерекурзивност овог скупа директно следи из чињенице да скуп $K = \{e \mid \varphi_e(x)\downarrow\}$ није рекурзиван:

$$e \in K \;\Leftrightarrow\; (e,e) \in H.$$

Из ове еквиваленције следи да ако би H био рекурзиван, такав би морао бити и скуп K.

Да скуп H није рекурзиван доказујемо и применом теореме рекурзије.

Петпоставимо да је H рекурзиван, тј. да је рекурзивна функција $\chi_H : \mathbb{N}^2 \to \mathbb{N}$,

$$\chi_H(e,x) = \begin{cases} 1, & (e,x) \in H, \\ 0, & (e,x) \notin H. \end{cases}$$

Нека је

$$f(e,x) = \begin{cases} \uparrow, & \chi_H(e,x) = 1, \\ 0, & \chi_H(e,x) = 0. \end{cases}$$

Из претпостављене рекурзивности функције χ_H следи да је f парцијално рекурзивна, па према теореми рекурзије постоји природан број e_0 такав да је $\varphi_{e_0}(x) \simeq f(e_0,x)$. Међутим, тада имамо:

- Ако $(e_0,x) \in H$, тада $f(e_0,x)\uparrow$ (према дефиницији функције f), а самим тим и $\varphi_{e_0}(x)\uparrow$, што значи да $(e_0,x) \notin H$. Контрадикција.
- Ако $(e_0,x) \notin H$, тада $f(e_0,x)\downarrow 0$ (према дефиницији функције f), а самим тим и $\varphi_{e_0}(x)\downarrow 0$, што значи да $(e_0,x) \in H$. Контрадикција.

Дакле, скуп H не може бити рекурзиван.

Примену теореме рекурзије илуструјемо и следећом конструкцијом.
Ако је $\mathbb{H}$ RM-програм који израчунава χ_H (и самим тим се зауставља за сваки улаз (e,x)), до контрадикције долазимо и конструкцијом следећег програма:

$\mathbb{D}$ =за улаз x :

1. Одреди (применом теореме рекурзије) сопствени код $[\![\mathbb{D}]\!]$
2. Позови $\mathbb{H}$ за улаз $([\![\mathbb{D}]\!], x)$
3. Ако $\mathbb{H}$ каже $([\![\mathbb{D}]\!], x) \in H$, онда дивергирај;
   а ако $\mathbb{H}$ каже $([\![\mathbb{D}]\!], x) \notin H$, онда врати излаз 0.

Рекурзивни оператори

Подсећамо, $\mathcal{F}_k$, $k \geqslant 1$, означава скуп свих k-арних парцијалних функција. Скуп $\mathcal{F}_k$ је природно уређен релацијом $\sqsubseteq$:

$(\mathcal{F}_k, \sqsubseteq)$ је комплетно уређење: најмање горње ограничиње растућег ланца јесте унија елемената тог ланца.

$$\varphi \sqsubseteq \psi \overset{\text{def}}{\Leftrightarrow} \forall x \in \mathbb{N}^k \forall y \in \mathbb{N}\ (\varphi(x) \simeq y \Rightarrow \psi(x) \simeq y).$$

Коначна функција јесте парцијална функција чији је домен коначан. Специјално, празна функција је коначна функција. Користећи неко ефективно кодирање уређених k-торки $\langle\cdot\rangle : \mathbb{N}^k \overset{1-1}{\underset{\text{на}}{\rightarrow}} \mathbb{N}$, једноставно дефинишимо кодирање свих k-арних коначних функција. Ако је θ нека k-арна коначна функција, њен кôд $\hat{\theta}$ је дефинисан на следећи начин:

$$\hat{\theta} = \begin{cases} \prod\limits_{\overline{x} \in \mathrm{dom}(\theta)} p_{\langle \overline{x} \rangle}^{\theta(\overline{x})+1}, & \mathrm{dom}(\theta) \neq \emptyset, \\ 0, & \mathrm{dom}(\theta) = \emptyset, \ (\theta \text{ је празна функција}). \end{cases}$$

Дефиниција 15. *Оператор $\Phi : \mathcal{F}_m \to \mathcal{F}_n$ је:*

- ***монотон*** *ако за све $\varphi, \psi \in \mathcal{F}_m$, из $\varphi \sqsubseteq \psi$ следи $\Phi(\varphi) \sqsubseteq \Phi(\psi)$;*
- ***непрекидан*** *ако за све $\varphi \in \mathcal{F}_m$, $\overline{x} \in \mathbb{N}^n$, $y \in \mathbb{N}$ важи: $\Phi(\varphi)(\overline{x}) \simeq y$ акко постоји коначна m-арна функција θ таква да је $\theta \sqsubseteq \varphi$ и $\Phi(\theta)(\overline{x}) \simeq y$;*

за сваки растући низ парцијалних функција $\varphi_0 \sqsubseteq \varphi_1 \sqsubseteq \varphi_2 \sqsubseteq \cdots$ важи $\Phi(\bigcup_i \varphi_i) = \bigcup_i \Phi(\varphi_i)$.

- ***ефективан над коначним функцијама*** *ако постоји $(n+1)$-арна парцијално рекурзивна функција ϕ таква да за све коначне m-арне функције θ и све $\overline{x} \in \mathbb{N}^n$ важи: $\Phi(\theta)(\overline{x}) \simeq \phi(\hat{\theta}, \overline{x})$.*

Дефиниција 16. *Оператор $\Phi : \mathcal{F}_m \to \mathcal{F}_n$ је* ***рекурзиван*** *ако постоји $(n+1)$-арна парцијално рекурзивна функција ϕ таква да за све $\varphi \in \mathcal{F}_m$, $\overline{x} \in \mathbb{N}^n$, $y \in \mathbb{N}$ важи: $\Phi(\varphi)(\overline{x}) \simeq y$ акко постоји коначна m-арна функција θ таква да је $\theta \sqsubseteq \varphi$ и $\phi(\hat{\theta}, \overline{x}) \simeq y$.*

Лема 12. *Сваки рекурзиван оператор је монотон, непрекидан и ефективан на коначним функцијама.*

Лема 13. *Оператор $\Phi : \mathcal{F}_m \to \mathcal{F}_n$ је рекурзиван акко важе следећа два услова*

(i) *Φ је непрекидан оператор и*

(ii) *функција дефинисана са:*

$\phi(\hat{\theta}, \overline{x}) \simeq \Phi(\theta)(\overline{x})$ *ако је θ коначна m-арна функција,*

$\phi(z, \overline{x})$ *није дефинисано када z није кôд коначне m-арне функције,*

јесте парцијално рекурзивна.

ПРИМЕР 56. Применом претходне леме, једноставно је доказати рекурзивност следећих оператора:

- (оператор дијагонализације) $\Phi : \mathcal{F}_2 \to \mathcal{F}_1$, $\Phi(f)(x) \simeq f(x,x)$ је рекурзиван: очигледно је непрекидан, а функција $\phi(\hat{\theta}, x) \simeq \theta(x,x)$ је парцијално рекурзивна;
- $\Phi : \mathcal{F}_1 \to \mathcal{F}_1$, $\Phi(f)(x) \simeq \sum_{y \leqslant x} f(y)$ је рекурзиван: непрекидан је и функција $\phi(\hat{\theta}, x) \simeq \sum_{y \leqslant x} \theta(y)$ је парцијално рекурзивна;
- ако је g нека унарна парцијално рекурзивна функција, онда је оператор $\Phi : \mathcal{F}_k \to \mathcal{F}_k$, $\Phi(f) \simeq g \circ f$ рекурзиван: непрекидан је и функција $\phi(\hat{\theta}, x) \simeq g(\theta(x))$ је парцијално рекурзивна;
- (μ-оператор) $\Phi : \mathcal{F}_{k+1} \to \mathcal{F}_k$, $\Phi(f)(\overline{x}) \simeq \mu y(f(\overline{x}, y) = 0)$ је рекурзиван: непрекидан је и функција $\phi(\hat{\theta}, x) \simeq \mu y(\theta(\overline{x}, y) = 0)$ је парцијално рекурзивна;
- ако је h нека $(n+1)$-арна парцијално рекурзивна функција, онда је оператор $\Phi : \mathcal{F}_{n+1} \to \mathcal{F}_{n+1}$,

$$\Phi(f)(x, \overline{y}) \simeq \begin{cases} 0, & h(x, \overline{y}) = 0, \\ f(x+1, \overline{y}) + 1, & h(x, \overline{y}) \downarrow \text{ и } h(x, \overline{y}) \neq 0. \end{cases}$$

рекурзиван.

Наредна теорема[160] тврди да сваки рекурзиван оператор представља једну ефективну операцију на скупу парцијално рекурзивних функција.

[160] Теорема је позната тако Мајхил-Шепердсон теорема.

Теорема 28. *Нека је $\Phi : \mathcal{F}_m \to \mathcal{F}_n$ рекурзиван оператор. Тада постоји рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да за све $e \in \mathbb{N}$ важи $\Phi(\varphi_e^{(m)}) = \varphi_{h(e)}^{(n)}$.*

ДОКАЗ. Из рекурзивности оператора Φ следи да постоји $(n+1)$-арна парцијално рекурзивна функција ϕ таква да за све $e \in \mathbb{N}$ ($\overline{x} \in \mathbb{N}^k$, $y \in \mathbb{N}$) важи:

$$\Phi(\varphi_e^{(m)})(\overline{x}) \simeq y \;\Leftrightarrow\; \text{постоји коначна функција } \theta \text{ т.д. } \theta \sqsubseteq \varphi_e^{(m)} \text{ и } \phi(\hat{\theta}, \overline{x}) \simeq y.$$

Кључно запажање у доказу ове теореме јесте чињеница да је следећи предикат рекурзивно набројив:

$$\Phi(\varphi_e^{(m)})(\overline{x}) \simeq y \Leftrightarrow \exists z\, (z \text{ је кôд неке коначне } m\text{-аре функције } \theta \text{ и } \theta \sqsubseteq \varphi_e^{(m)} \text{ и } \phi(z, \overline{x}) \simeq y)$$

Одатле следи да је функција $(e, \overline{x}) \mapsto \Phi(\varphi_e^{(m)})(\overline{x})$ парцијално рекурзивна. Према теореми параметризације постоји рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да је $\varphi_{h(e)}^{(n)}(\overline{x}) \simeq \Phi(\varphi_e^{(m)})(\overline{x})$. □

НАПОМЕНА. Тотална функција $h : \mathbb{N} \to \mathbb{N}$ је екстензионална ако за све $a, b \in \mathbb{N}$, из $\varphi_a = \varphi_b$ следи да је $\varphi_{h(a)} = \varphi_{h(b)}$. Може се доказати да за сваку рекурзивну екстензионалну функцију $h : \mathbb{N} \to \mathbb{N}$ постоји јединствени рекурзивни оператор $\Phi : \mathcal{F}_1 \to \mathcal{F}_1$ такав да је $\Phi(\varphi_e) = \varphi_{h(e)}$, за све $e \in \mathbb{N}$.

Теорема 29. [***Прва теорема рекурзије***] *Нека је* $\Phi : \mathcal{F}_k \to \mathcal{F}_k$ *рекурзиван оператор. Тада постоји јединствена парцијално рекурзивна* k*-арна функција* f_Φ *која је најмања (у смислу* $\sqsubseteq$*) фиксна тачка оператора* Φ*, тј. важи:*

(i) $\Phi(f_\Phi) = f_\Phi$*;*

(ii) *Ако је* $\Phi(g) = g$*, онда је* $f_\Phi \sqsubseteq g$.

Доказ. Дефинишимо низ функција (f_n) на следећи начин:

$$f_0 = \emptyset \ (f_0 \text{ је празна функција}),$$
$$f_{n+1} = \Phi(f_n).$$

Из $f_0 \sqsubseteq f_1$, због монотоности оператора Φ следи да је $f_n \sqsubseteq f_{n+1}$, за све $n \in \mathbb{N}$. Нека је $f_\Phi = \cup_{n\in\mathbb{N}} f_n$. Дакле,

$$f_\Phi(\overline{x}) \simeq y \ \Leftrightarrow \ \exists n \in \mathbb{N} \ (f_n(\overline{x}) \simeq y).$$

Показаћемо да је f_Φ фиксна тачка оператора Φ. За свако $n \in \mathbb{N}$, $f_n \sqsubseteq f_\Phi$, одакле следи да је $f_{n+1} = \Phi(f_n) \sqsubseteq \Phi(f_\Phi)$. Дакле, $f_\Phi \sqsubseteq \Phi(f_\Phi)$. Да бисмо доказали обратно, претпоставимо да је $\Phi(f_\Phi)(\overline{x}) \simeq y$. Тада постоји коначна функција $\theta \sqsubseteq f_\Phi$ таква да је $\Phi(\theta)(\overline{x}) \simeq y$. Нека је $n \in \mathbb{N}$ такав да је $\theta \sqsubseteq f_n$. Из непрекидности оператора Φ следи да је $\Phi(f_n)(\overline{x}) \simeq y$, тј. $f_{n+1}(\overline{x}) \simeq y$. Дакле, $f_\Phi(\overline{x}) \simeq y$, одакле следи да је $\Phi(f_\Phi) \sqsubseteq f_\Phi$. Тиме је завршен доказ једнакости $\Phi(f_\Phi) = f_\Phi$.

Ако је $\Phi(g) = g$, једноставно се индукцијом показује да је $f_n \sqsubseteq g$, за свако n, одакле следи да је $f_\Phi \sqsubseteq g$.

Остаје још да докажемо да је f_Φ парцијално рекурзивна функција. Према претходној теореми, постоји рекурзивна функција $h : \mathbb{N} \to \mathbb{N}$ таква да је $\Phi(\varphi_e^{(k)}) = \varphi_{h(e)}^{(k)}$, за све $e \in \mathbb{N}$. Нека је e_0 произвољан индекс празне функције и $e_{n+1} = h(e_n)$, $n \in \mathbb{N}$. Очигледно, e је рекурзивна функција. Штавише, једноставно се уочава да је $f_n = \varphi_{e_n}^{(k)}$. Одавде следи еквиваленција:

$$f_\Phi(\overline{x}) \simeq y \ \Leftrightarrow \ \exists n \ (\varphi_{e_n}^{(k)}(\overline{x}) \simeq y).$$

Лева страна еквиваленције одређује један рекурзивно набројив предикат, одакле следи да је f_Φ парцијално рекурзивна функција. □

Процедура израчунавања вредности $f_\Phi(\overline{x})$ могла би се неформално описати на следећи начин:
1. $i := 0$
2. за све $j \leqslant i$, изврши $(i+1)$ корака програма e_j за улаз $\overline{x}$:
- ако се у неком случају достигне завршна конфигурација са излазом y, заустави се и врати излаз y;
- иначе, израчунај $e_{i+1} = h(e_i)$, стави $i := i+1$ и врати се на извршавање корака 2.

Последица 8. *Ако је најмања фиксна тачка рекурзивног оператора тотална функција, онда је та функција уједно и једина фиксна тачка оператора.*

Пример 57. Одредимо најмање фиксне тачке неколико рекурзивних оператора.

Нека је $\Phi : \mathcal{F}_1 \to \mathcal{F}_1$,

$$\begin{aligned}&\Phi(f)(0) = 1,\\ &\Phi(f)(x+1) \simeq f(x+2).\end{aligned}$$

Најмања фиксна тачка оператора Φ јесте функција

$$\begin{cases} f_\Phi(0) = 1, \\ f_\Phi(x+1) \text{ није дефинисано.} \end{cases}$$

Остале фиксне тачке оператора Φ су облика:

$$\begin{cases} f_\Phi(0) = 1, \\ f_\Phi(x+1) = c. \end{cases}$$

Нека је h нека $(n+1)$-арна парцијално рекурзивна функција, и $\Phi : \mathcal{F}_{n+1} \to \mathcal{F}_{n+1}$,

$$\Phi(f)(x,\overline{y}) \simeq \begin{cases} 0, & h(x,\overline{y}) = 0, \\ f(x+1,\overline{y})+1, & h(x,\overline{y})\downarrow \text{ и } h(x,\overline{y}) \neq 0. \end{cases}$$

рекурзиван оператор дефинисан у претходном примеру. Најмања фиксна тачка f_Φ овог оператора јесте парцијално рекурзивна функција која задовољава следеће услове:

$$f_\Phi(x,\overline{y}) \simeq \begin{cases} 0, & h(x,\overline{y}) = 0, \\ f(x+1,\overline{y})+1, & h(x,\overline{y})\downarrow \text{ и } h(x,\overline{y}) \neq 0, \\ \uparrow, & \text{иначе.} \end{cases}$$

Није тешко проверити да је $f_\Phi(x,\overline{y}) \simeq \mu z(h(x+z,\overline{y}) = 0)$. Заиста, ако је $\mu z(h(x+z,\overline{y}) = 0) = m$, онда је $h(x+z,\overline{y})$ дефинисано и различито од нуле за све $z < m$, и важи $h(x+m,\overline{y}) = 0$, одакле следи

$$\begin{aligned} f_\Phi(x,\overline{y}) &= f_\Phi(x+1,\overline{y})+1 = \cdots = f_\Phi(x+z,\overline{y})+z \ (z \leqslant m) \\ &= f_\Phi(x+m,\overline{y})+m = 0+m = m. \end{aligned}$$

Претпоставимо сада да је $f_\Phi(x,\overline{y}) = m$. Тада је

$$m = f_\Phi(x,\overline{y}) = f_\Phi(x+1,\overline{y})+1 = \cdots = f_\Phi(x+m,\overline{y})+m,$$

одакле следи да је $h(x+z,\overline{y})$ дефинисано и различито од нуле за све $z < m$, па је $f_\Phi(x+m,\overline{y}) = 0$, а самим тим је $h(x+m,\overline{y}) = 0$. Дакле, $m = \mu z(h(x+z,\overline{y}) = 0)$.

Прва теорема рекурзије има значајне примене на проблеме који се односе на семантику програмских језика.

m-СВОДЉИВОСТ

Сводљивост једног проблема на други је природан метод класификације неодлучивих проблема. Основна идеја је веома једноставна. На пример, ако желимо да покажемо да је неки проблем B неодлучив, довољно је да нађемо неки други проблем A такав да важи:

1) A је неодлучив;

2) одлучивост проблема B имплицира одлучивост проблема A.

Из ова два услова директно закључујемо да је проблем B неодлучив.

Дефиниција 17. *Нека су A и B подскупови од $\mathbb{N}$. Скуп A је* **m-сводљив** *на B, у ознаци $A \leq_{\mathrm{m}} B$, ако постоји унарна рекурзивна функција f таква да је за свако $x \in \mathbb{N}$ важи:*

$$(*) \qquad x \in A \Leftrightarrow f(x) \in B.$$

Индекс m, у ознаци $\leq_{\mathrm{m}}$, скраћује израз „many to one“ који изражава само то да функција f не мора бити 1-1.

Интуитивно, ако је $A \leq_{\mathrm{m}} B$, проблем '$x \in B$' *није лакши* од проблема '$x \in A$'.

Ако је $A \leq_{\mathrm{m}} B$ и f је унарна рекурзивна функција којом се оправдава ова m-сводљивост, писаћемо $f : A \leq_{\mathrm{m}} B$.

Лема 14. *1) За сваки скуп $A \subseteq \mathbb{N}$, $A \leq_{\mathrm{m}} A$.*

2) За све $A, B, C \subseteq \mathbb{N}$, ако је $A \leq_{\mathrm{m}} B$ и $B \leq_{\mathrm{m}} C$, онда је $A \leq_{\mathrm{m}} C$.

Доказ. 1) $\mathrm{id}_{\mathbb{N}} : A \leq_{\mathrm{m}} A$.
2) Ако $f : A \leq_{\mathrm{m}} B$ и $g : B \leq_{\mathrm{m}} C$, онда $g \circ f : A \leq_{\mathrm{m}} C$. □

Лема 15. *Ако је $K \leq_{\mathrm{m}} A$, онда A није рекурзиван.*

Доказ. Нека је f рекурзивна функција таква да $f : K \leq_{\mathrm{m}} A$, тј. за све $x \in \mathbb{N}$, $x \in K \Leftrightarrow f(x) \in A$. Ако би A био рекурзиван скуп, тј. χ_A рекурзивна функција, онда би из једнакости $\chi_K = \chi_A \circ f$ следило и да је χ_K рекурзивна функција, што није тачно. □

Пример 58. Докажимо да скуп $E = \{e \mid \mathrm{dom}(\varphi_e) \neq \emptyset\}$ није рекурзиван.

Доказаћемо да се K може свести на E, тј. да проблем '$x \in E$' није лакши од проблема '$x \in K$'. Функција

$$f(e,x) = \begin{cases} 0, & e \in K \text{ (тј. } \varphi_e(e)\downarrow), \\ \uparrow, & e \notin K \text{ (тј. } \varphi_e(e)\uparrow), \end{cases}$$

је парцијално рекурзивна. Нека је e_0 индекс функције f и $h : \mathbb{N} \to \mathbb{N}$ примитивно рекурзивна функција дефинисана са $h(e) \overset{\mathrm{def}}{=} s^1_1(e_0, e)$. Тада је:

$$\varphi_{h(e)}(x) \simeq \varphi_{s^1_1(e_0,e)}(x) \simeq \varphi^{(2)}_{e_0}(e,x) \simeq f(e,x), \text{ за свако } x \in \mathbb{N}.$$

Ако $e \in K$, онда је $\varphi_{h(e)} = \mathbf{0}$, па $h(e) \in E$. Ако $e \notin K$, онда је $\varphi_{h(e)}$ празна функција, па $h(e) \notin E$. Дакле: $e \in K \Leftrightarrow h(e) \in E$. Из ове еквиваленције следи да E не може бити рекурзиван, јер би такав био и скуп K.

Други начин. Поређења ради, доказаћемо да E није рекурзиван и применом теореме рекурзије. Претпоставимо да је E рекурзиван скуп. Тада је функција

$$f(e,x) = \begin{cases} 0, & e \notin E, \\ \uparrow, & e \in E, \end{cases}$$

парцијално рекурзивна, па према теореми рекурзије постоји e_0 такав да је $\varphi_{e_0}(x) \simeq f(e_0,x)$, за све $x \in \mathbb{N}$. Међутим, одавде закључујемо:

- ако $e_0 \notin E$, онда је $\varphi_{e_0} = \mathbf{0}$, што значи да је φ_{e_0} тотална функција, а то није могуће, као и
- ако $e_0 \in E$, φ_{e_0} је празна функција, па није тотална, што је такође немогуће.

Ако претпоставимо да постоји RM-програм $\mathbb{E}$ који одлучује E, до контрадикције долазимо и конструкцијом следећег програма:

$\mathbb{D}$ =за улаз x :
1. Одреди (применом теореме рекурзије) сопствени код $[\![\mathbb{D}]\!]$
2. Позови $\mathbb{E}$ за улаз $([\![\mathbb{D}]\!], x)$
3. Ако $\mathbb{E}$ каже $([\![\mathbb{D}]\!], x) \in E$, онда дивергирај;
   а ако $\mathbb{E}$ каже $([\![\mathbb{D}]\!], x) \notin E$, онда врати излаз 0.

Пример 59. Нека је $\mathbf{Tot} = \{e \mid \mathrm{dom}(\varphi_e) = \mathbb{N}\}$ и $\mathbf{Nul} = \{e \mid \varphi_e = \mathbf{0}\}$. Докажимо да је $\mathbf{Tot} \leq_{\mathrm{m}} \mathbf{Nul}$.

Према теореми параметризације постоји унарна рекурзивна функција k таква да је $\varphi_{k(e)} = \mathbf{0} \circ \varphi_e$, за свако $e \in \mathbb{N}$. Тада важи:

$$e \in \mathbf{Tot} \Leftrightarrow k(e) \in \mathbf{Nul},$$

тј. $k : \mathbf{Tot} \leq_{\mathrm{m}} \mathbf{Nul}$.

Наводимо још једну важну карактеризацију рекурзивно набројивих подскупова од $\mathbb{N}$, према којој се проблем припадања било ком рекурзивно набројивом подскупу од $\mathbb{N}$ може ефективно свести на проблем припадања скупу K. То значи да је проблем K најтежи међу свим рекурзивно набројивим скуповима.

Теорема 30. *За сваки рекурзивно набројив скуп A важи $A \leq_{\mathrm{m}} K$.*

Доказ. Из теореме параметризације следи да постоји рекурзивна функција $s_1^1 : \mathbb{N}^2 \to \mathbb{N}$ таква да је за све $e, x, y \in \mathbb{N}$:

$$\varphi_e^{(2)}(x,y) \simeq \varphi_{s_1^1(e,x)}(y).$$

Нека је $A = \mathrm{dom}(\varphi_{e_0})$, за неко e_0. Дефинишимо бинарну функцију g на следећи начин

$$g(x,y) \simeq \varphi_{e_0}(x).$$

Функција g је парцијално рекурзивна, па постоји природан број c такав да је $g = \varphi_c^{(2)}$. Нека је $f(x) = s_1^1(c,x)$, $x \in \mathbb{N}$. Тада је за

свако $x \in \mathbb{N}$:

$$\begin{aligned} x \in A &\Leftrightarrow \varphi_{e_0}(x)\downarrow \\ &\Leftrightarrow g(x, s_1^1(c,x))\downarrow \\ &\Leftrightarrow \varphi_c^{(2)}(x, s_1^1(c,x))\downarrow \\ &\Leftrightarrow \varphi_{s_1^1(c,x)}(s_1^1(c,x))\downarrow \\ &\Leftrightarrow s_1^1(c,x) \in K \\ &\Leftrightarrow f(x) \in K. \end{aligned}$$

□

Рекурзивни скупови, различити од $\emptyset$ и $\mathbb{N}$, представљају најлакше проблеме у смислу m-сводљивости. Штавише, свака два оваква скупа су m-еквивалентна, тј. сваки од њих је m-сводљив на други.

Теорема 31. *Нека су $A, B \subseteq \mathbb{N}$ произвољни скупови.*

1) Ако је A рекурзиван и $B \leq_{\mathrm{m}} A$, онда је и B рекурзиван.

2) Ако је A рекурзиван и $B \neq \emptyset, \mathbb{N}$, онда је $A \leq_{\mathrm{m}} B$.

3) Ако је A рекурзивно набројив скуп и $B \leq_{\mathrm{m}} A$, онда је и B рекурзивно набројив.

Доказ. 1) Нека је A рекурзиван скуп и $f : B \leq_{\mathrm{m}} A$. Тада је χ_A рекурзивна функција и важи $\chi_B = \chi_A \circ f$, одакле следи да је и χ_B рекурзивна функција, односно да је B рекурзиван скуп.
2) Претпоставимо да је A рекурзиван скуп и $B \neq \emptyset, \mathbb{N}$. Нека је b произвољан елемент из B и c произвољан елемент из $B^{\complement}$. Дефинишимо функцију $f : \mathbb{N} \to \mathbb{N}$ са:

$$f(x) = \begin{cases} b, & x \in A, \\ c, & x \in A^{\complement}. \end{cases}$$

Како је A рекурзиван скуп, функција f је рекурзивна и важи:

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

Дакле, $f : A \leq_{\mathrm{m}} B$.
3) Нека је A рекурзивно набројив скуп и $f : B \leq_{\mathrm{m}} A$. Постоји парцијално рекурзивна функција g таква да је $A = \mathrm{dom}(g)$. Функција $g \circ f$ је такође парцијално рекурзивна и $\mathrm{dom}(g \circ f) = B$, што значи да је B рекурзивно набројив. □

Неодлучиви проблеми

Рајсова теорема

Уопштено говорећи, Рајсова теорема показује да су неодлучива сва нетривијална својства програма.

Дефиниција 18. *Скуп $A \subseteq \mathbb{N}$ је екстензионалан ако за све $a, e \in \mathbb{N}$:*

$$\text{из } a \in A \text{ и } \varphi_e = \varphi_a \text{ следи } e \in A.$$

Скуп $A \subseteq \mathbb{N}$ је нетривијалан ако је $A \neq \emptyset$ и $A \neq \mathbb{N}$; у супротном је тривијалан.

Теорема 32. [*Рајсова теорема*] *Сваки нетривијалан екстензионалан скуп није рекурзиван.*

Доказ. **Први начин** (примена теореме рекурзије).

Нека је $A \subseteq \mathbb{N}$ нетривијалан екстензионалан скуп; $a \in A$ и $b \in \mathbb{N} \setminus A$. Претпоставимо да је A рекурзиван скуп. Тада је функција

$$f(e,x) = \begin{cases} \varphi_b(x), & e \in A, \\ \varphi_a(x), & e \notin A, \end{cases}$$

парцијално рекурзивна. Према теореми рекурзије, постоји $e_0 \in \mathbb{N}$ такав да је $\varphi_{e_0}(x) \simeq f(e_0, x)$, $x \in \mathbb{N}$.

$f(e,x) \simeq \Phi_U(b,x) \cdot \chi_A(x) + \Phi_U(a,x) \cdot \chi_{\mathbb{N}\setminus A}(x)$

- Ако $e_0 \in A$, онда је $\varphi_{e_0} = \varphi_b$, а како је A екстензионалан и $b \notin A$, следи да $e_0 \notin A$. Контрадикција.
- Ако $e_0 \notin A$, онда је $\varphi_{e_0} = \varphi_a$, а како је A екстензионалан и $a \in A$, следи да $e_0 \in A$. Контрадикција.

Дакле, A није рекурзиван скуп.

Други начин (примена теореме параметризације).

Нека је $e_\emptyset$ индекс празне функције.

Ако $e_\emptyset \in A$, нека је a било који елемент скупа $A^{\complement}$.

Ако $e_\emptyset \notin A$, нека је a било који елемент скупа A.

Функција

$$f(e,x) = \begin{cases} \varphi_a(x), & e \in K, \\ \uparrow, & e \notin K, \end{cases}$$

јесте парцијално рекурзивна, и нека је e_0 њен индекс. Применом теореме параметризације дефинишимо примитивно рекурзивну функцију $h : \mathbb{N} \to \mathbb{N}$, $h(e) \overset{\text{def}}{=} s_1^1(e_0, e)$. Тада је

$f(e,x) \simeq \Phi_U(a,x) \cdot \mathbf{1}(\Phi_U(e,e))$

$$\varphi_{h(e)}(x) \simeq \varphi_{s_1^1(e_0,e)}(x) \simeq \varphi_{e_0}^{(2)}(e,x) \simeq f(e,x), \quad \text{за свако } x \in \mathbb{N}.$$

Размотримо случај када $e_\emptyset \in A$. Тада $a \notin A$.
Ако $e \in K$, онда је $\varphi_{h(e)} = \varphi_a$, па из екстензионалности скупа A следи да $h(e) \notin A$.
Ако $e \notin K$, онда је $\varphi_{h(e)}$ празна функција, па из екстензионалности скупа A и чињенице да $e_\emptyset \in A$ следи да $h(e) \in A$.
Дакле: $e \in K \Leftrightarrow h(e) \in A^{\complement}$.

Размотримо случај када $e_\emptyset \notin A$. Тада $a \in A$.
Ако $e \in K$, онда је $\varphi_{h(e)} = \varphi_a$, па из екстензионалности скупа A следи да $h(e) \in A$.
Ако $e \notin K$, онда је $\varphi_{h(e)}$ празна функција, па из екстензионалности скупа A и чињенице да $e_\emptyset \notin A$ следи да $h(e) \notin A$.
Дакле: $e \in K \Leftrightarrow h(e) \in A$.

У оба случаја закључујемо да A не може бити рекурзиван, јер би онда такав морао бити скуп K. □

Последица 9. *Нека је $\mathcal{A}$ било који скуп унарних парцијално рекурзивних функција. Скуп $I(\mathcal{A}) = \{e \mid \varphi_e \in \mathcal{A}\}$ је рекурзиван акко је $\mathcal{A}$ тривијалан скуп, тј. $\mathcal{A} = \emptyset$ или $\mathcal{A}$ садржи све унарне парцијално рекурзивне функције.*

Доказ. ($\Leftarrow$) Ако је $\mathcal{A} = \emptyset$, онда је $I(\mathcal{A}) = \emptyset$. Ако $\mathcal{A}$ садржи све унарне парцијално рекурзивне функције, онда је $I(\mathcal{A}) = \mathbb{N}$. У оба случаја тврђење важи, јер су скупови $\emptyset$ и $\mathbb{N}$ рекурзивни.
($\Rightarrow$) Скуп $I(\mathcal{A})$ је сигурно екстензионалан. Ако је $I(\mathcal{A})$ и рекурзиван, према Рајсовој теореми мора бити тривијалан: $I(\mathcal{A}) = \emptyset$ или $I(\mathcal{A}) = \mathbb{N}$. Дакле, $\mathcal{A} = \emptyset$ или $\mathcal{A}$ садржи све унарне парцијално рекурзивне функције. □

Неодлучивост проблема речи

Алфабет је коначан низ симбола; реч на алфабету Σ је сваки коначан низ симбола из Σ. Уређен пар речи (u, v) називаћемо Σ-правилом и означавати $u \to v$. Свако Σ-правило описује једну трансформацију скупа свих речи Σ^*. Ако реч $w \in \Sigma^*$ садржи u као подреч, одн. ако је $w = w_1 u w_2$, за неке $w_1, w_2 \in \Sigma^*$, онда применом правила $P: u \to v$ на реч w изводимо нову реч $w' = w_1 v w_2$ (w' је добијено из w заменом подречи u речју v) и пишемо $w \to_P w'$.

Коначан скуп Σ-правила називаћемо Σ-процесом. Ако је $\mathcal{P}$ неки Σ-процес и $w, w' \in \Sigma^*$, тада:

- $w \to_{\mathcal{P}} w'$ значи да је $w \to_P w'$ за неко Σ-правило P из $\mathcal{P}$;
- $w \to_{\mathcal{P}}^* w'$ значи да је $w = w'$ или постоји коначан низ речи $w_1, \ldots, w_k \in \Sigma^*$, $k > 1$, таквих да је $w = w_1$, $w' = w_k$ и $w_i \to_{\mathcal{P}} w_{i+1}$, $1 \leqslant i < k$. Низ $w_1, \ldots, w_k$ називамо и извођењем речи w' из речи w.

ПРИМЕР 60. Нека је $\Sigma = \{a, b\}$ и $\mathcal{P} = \{ab \to aa, ba \to bb\}$. Тада је, на пример, $aba \to_{\mathcal{P}} abb \to_{\mathcal{P}} aab \to_{\mathcal{P}} aaa$, па $aba \to^*_{\mathcal{P}} aaa$. Није тешко приметити да се из aaa не може, применом правила из $\mathcal{P}$ извести реч bbb, што записујемо $aaa \not\to^*_{\mathcal{P}} bbb$.

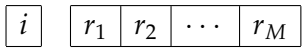
Проблем речи за алфабет Σ: да ли постоји алгоритам који за задати Σ-процес $\mathcal{P}$ и речи $w, w' \in \Sigma$ испитује (одлучује) да ли $w \to^*_{\mathcal{P}} w'$ или $w \not\to^*_{\mathcal{P}} w'$?

Показаћемо, за погодно изабран алфабет Σ, да не постоји алгоритам који захтева проблем речи, чак и за један фиксирани Σ-процес $\mathcal{P}$ и фиксирану реч w'. Другим речима, конструисаћемо Σ-процес $\mathcal{P}$ и изабрати једну реч w' тако да *не постоји алгоритам који за задату реч w испитује да $w \to^*_{\mathcal{P}} w'$ или $w \not\to^*_{\mathcal{P}} w'$.*

Изабраћемо алфабет Σ који је погодан за описивање извршавања RM-програма $\mathbb{P}$: $I_1, \ldots, I_N$ који израчунава функцију $x \mapsto \Phi_U(x, x)$. Претпоставимо да је $\mathbb{P}$ у стандардној форми и да је $M = \|\mathbb{P}\| + 1$. Нека је

$$\Sigma = \{b, a, r_1, \ldots, r_M, p_1, \ldots, p_N, p_{N+1}, q_1, \ldots, q_N, e_1, \ldots, e_N\}.$$

Сваку конфигурацију, током извршавања програма $\mathbb{P}$,



описаћемо као реч

$$bp_i r_1 a^{r_1} r_2 a^{r_2} \cdots r_M a^{r_M},$$

где a^r означава r узастопних појављивања симбола a, $\overbrace{aa\cdots a}^{r \text{ пута}}$, уз договор да је a^0 празна реч. Специјално, почетној конфигурацији за улаз $x \in \mathbb{N}$ одговара реч $bp_1 r_1 a^x r_2 r_3 \cdots r_{M-1} r_M$, коју ћемо краће означавати w_x.

Σ-процес $\mathcal{P}$ конструишемо на основу програма $\mathbb{P}$, тако што за сваку инструкцију I_i, $1 \leqslant i \leqslant N$, додајемо Σ-правила којима се симулира извршавање те инструкције.

Ако је I_i инструкција $R_k^+ \mid \ell$, онда додајемо следећа правила:

$p_i r_j \to r_j p_i$, $1 \leqslant j < k$,
$p_i a \to a p_i$,

Ако је $k > 1$ додајемо најпре правила помоћу којих се из неке речи која описује конфигурацију изводи реч у којој је симбол p_i постављен испред r_k; наравно, ако је $k = 1$, онда је већ p_i већ постављено испред r_1.

$p_i r_k \to q_i r_k a$,

Додајемо правило које одговара повећавању садржаја регистра R_k за 1. Увођење слова q_i (уместо p_i) означава чињеницу да је извршена акција на регистру R_k коју налаже инструкција I_i.

$r_j q_i \to q_i r_j$, $1 \leqslant j < k$,
$a q_i \to q_i a$,

Враћамо (само ако је $k > 1$) слово q_i до слова b.

$b q_i \to b p_\ell$.

Мењамо q_i словом p_ℓ.

Ако је I_i инструкција $R_k^- \mid \ell_0, \ell_1$, онда додајемо правила:

$p_i r_j \to r_j p_i$, $1 \leqslant j < k$,
$p_i a \to a p_i$.
$p_i r_k a \to q_i r_k$,
$p_i r_k r_{k+1} \to e_i r_k r_{k+1}$,

Додајемо правило које одговара акцији на регистру R_k: ако у R_k није уписана нула, онда се садржај умањује за 1, и та акција се „памти“ словом q_i; а ако је у R_k уписана нула, онда се садржај не мења, и тај случај се „памти“ словом e_i.

$r_j q_i \to q_i r_j$, $1 \leqslant j < k$,
$a q_i \to q_i a$,
$r_j e_i \to e_i r_j$, $1 \leqslant j < k$,
$a e_i \to e_i a$,
$b q_i \to b p_{\ell_1}$,
$b e_i \to b p_{\ell_0}$.

Најзад, додајемо и правила

$$p_{N+1} r_i \to p_{N+1},\ p_{N+1} a \to p_{N+1},$$

којима се реч, која одговара завршној конфигурацији (ако се она уопште достиже) израчунавања $\mathbb{P}(x)$, трансформише у $b p_{N+1}$.

Није тешко закључити да за сваки природан број x важи:

$w_x \to_{\mathcal{P}}^* b p_{N+1}$ акко $\Phi_U(x,x)\downarrow$, одн. $w_x \to_{\mathcal{P}}^* b p_{N+1}$ акко $x \in K$.

Будући да скуп K није рекурзиван, не постоји алгоритам који се тражи у проблему речи.

Неодлучивост проблема ваљаности

Проблем ваљаности за сигнатуру $\mathcal{L}$ логике првог реда: да ли постоји алгоритам који за задату $\mathcal{L}$-реченицу σ испитује (одлучује) да ли је σ ваљана (тачна у свим моделима) или није, тј. постоји модел у којем није тачна?

Показаћемо, за погодно изабрану сигнатуру $\mathcal{L}$, да не постоји алгоритам који захтева проблем ваљаности.

Изабраћемо сигнатуру $\mathcal{L}$ која је погодна за описивање извршавања RM-програма $\mathbb{P}$: $I_1, \ldots, I_N$ који израчунава функцију $x \mapsto \Phi_U(x,x)$. Претпоставимо да је $\mathbb{P}$ у стандардној форми и да је $M = \|\mathbb{P}\| + 1$. Нека $\mathcal{L}$ садржи:

- један симбол константе $\underline{0}$,
- унарни операцијски симбол $'$ и
- $d+1$-арни релацијски симбол R.

За свако $n \in \mathbb{N}$, терм $\underline{0}\overbrace{'\cdots'}^{n \text{ пута}}$ (тзв. *нумерал*) означићемо $\underline{n}$.

За сваку инструкцију $\mathbb{I}_i$ програма $\mathbb{P}$ саставићемо реченицу σ_i која изражава значење те инструкције.

Ако је I_i инструкција $R_k^+ \mid \ell$, онда је σ_i реченица

$$\forall x_1 \ldots \forall x_d (R(\underline{i}, x_1, \ldots, x_k, \ldots, x_d) \Rightarrow R(\underline{\ell}, x_1, \ldots, x_k', \ldots, x_d)).$$

Ако је I_i инструкција $R_k^- \mid \ell_0, \ell_1$, онда је σ_i реченица

$$\begin{aligned}\forall x_1 \ldots \forall x_d (R(\underline{i}, x_1, \ldots, \underline{0}, \ldots, x_d) &\Rightarrow R(\underline{\ell_0}, x_1, \ldots, \underline{0}, \ldots, x_d) \wedge \\ \wedge\ R(\underline{i}, x_1, \ldots, x_k', \ldots, x_d) &\Rightarrow R(\underline{\ell_1}, x_1, \ldots, x_k, \ldots, x_d)).\end{aligned}$$

Нека је σ_0 реченица

$$\forall x \forall y ((x' = y' \Rightarrow x = y) \wedge x' \neq \underline{0}).$$

За сваки природан број $n \in \mathbb{N}$ нека је σ_n реченица

$$\sigma_0 \wedge \sigma_1 \wedge \cdots \wedge \sigma_N \wedge R(\underline{1}, \underline{n}, \underline{0}, \ldots, \underline{0}) \Rightarrow \exists x_1 \cdots \exists x_d R(\underline{N+1}, x_1, \ldots, x_d).$$

Није тешко закључити да за сваки природан број n важи:

$$\mathbb{P}(n) \downarrow \Leftrightarrow \models \sigma_n, \text{ одн. } n \in K \Leftrightarrow \models \sigma_n.$$

Будући да скуп K није одлучив, исто важи и за проблем ваљаности.

X *Хилбертов проблем*

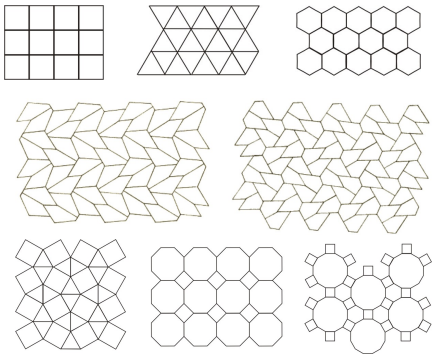
Одељак завршавамо кратким коментаром о неодлучивости X Хилбертовог проблема: да ли постоји алгоритам који за сваки полином $P(x_1, \ldots, x_k)$, чији су коефицијенти природни бројеви, одлучује да ли једначина $P(x_1, \ldots, x_k) = 0$ има решења у $\mathbb{N}^k$ или нема?

Нећемо доказивати да овај алгоритам не постоји, већ само укратко наводимо суштину доказа. Скуп $A \subseteq \mathbb{N}^k$ је *диофантовски* уколико постоји полином $P(x_1, \ldots, x_k, y_1, \ldots, y_m)$ такав да је

$$A = \{\vec{x} \in \mathbb{N}^k \mid \exists y_1 \ldots y_m \, P(\vec{x}, \vec{y}) = 0.\}$$

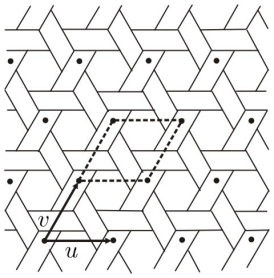
Очигледно је сваки диофантовски скуп рекурзивно набројив. Међутим, тачно је и обрнуто, што је главна тврдња Матијасевичеве теореме: *сваки рекурзивно набројив скуп је диофантовски.* Из ове теореме директно следи негативно решење X Хилбертовог проблема.

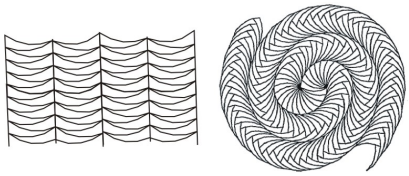
Неодлучивост проблема поплочавања



Познато је да се раван може *поплочати*, тј. покрити без преклапања и празнина, квадратима, једнакостраничним троугловима и правилним шестоугловима, а да се не може поплочати, на пример, правилним петоугловима. Наравно, постоје (неправилни) петоуглови којима се може поплочати раван. Разматрање поплочавања равни прилично се компликује повећавањем броја облика.

Сва поплочавања равни приказана на сликама десно јесу *периодична*. Грубо говорећи, то значи да постоје бар два различита правца у поплочаној равни и на сваком од њих бесконачно много тачака из којих можемо посматрати поплочану раван а да оно што видимо буде један те исти шаблон, односно да нам поплочана раван изгледа потпуно исто када је посматрамо из било које од поменутих тачака. Математичким језиком, поплочавање је периодично ако постоје две независне транслације равни које поплочавање преводе у себе. За свако периодично поплочавање равни постоји тзв. *паралелограм периода* одређен векторима транслација које то поплочавање преводе у себе (слика десно). Постоје плочице којима се раван може поплочавати и периодично и непериодично.



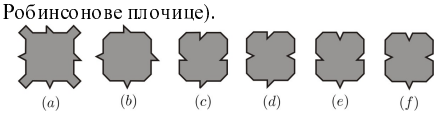


Постоји ли коначан скуп дозвољених облика који се могу редјати само непериодично? Прецизније, постоји ли коначан скуп плочица којима се може поплочати раван, али свако од могућих поплочавања није периодично? Такав скуп плочица се назива *апериодичан скуп плочица*. Одговор на претходно питање је потврдан – постоји апериодичан скуп плочица.

Како и зашто су откривени апериодични скупови плочица? Хао Ванг, кинески логичар, 1961. године формулисао је проблем: Постоји ли ефективан поступак за решавање проблема поплочавања, тј. постоји ли неки алгоритам којим се може утврдити да ли дати скуп многоугаоних плочица може поплочати раван? Ванг је доказао да овакав алгоритам постоји ако не постоји апериодичан скуп плочица (тј. за сваки скуп плочица који поплочава раван бар једно поплочавање је периодично). Постојање апериодичног скупа плочица потврдио је Робинсон 1971. године, тако што је доказао да је проблем поплочавања неодлучив.

Кључна идеја доказа неодлучивости проблема поплочавања равни јесте симулација Тјурингових машина помоћу плочица.

На наредној слици је приказан један апериодичан скуп плочица (приказане су тзв. Робинсонове плочице).

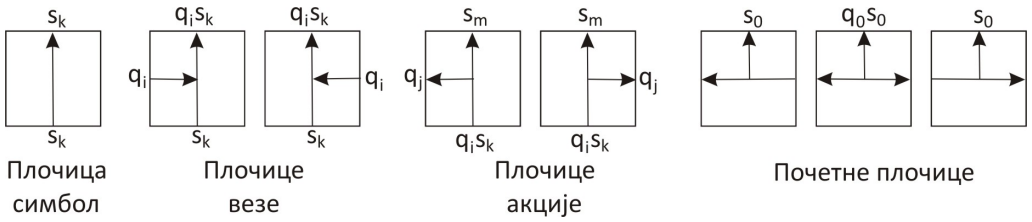


Н. Икодиновић, *Апериодични скупови плочица*, Тангента 66/2, 2011/2012

Симулација је занимљива сама по себи јер омогућава да паковање плочица замишљамо као модел израчунљивости.

Симулираћемо рад произвољне Тјурингове машине на празној траци, јер: не постоји алгоритам који одлучује да ли се произвољна Тјурингова машина зауставља ако започне рад на празној траци. Без губитка општости, скицираћемо доказ неодлучивости *проблема поплочавања полуравни са почетним захтевима.* За симулацију Тјурингових машина на празној траци користимо следеће плочице.

Овај закључак једноставно следи из варијанте Рајсове теореме формулисане у контексту Тјурингових машина.

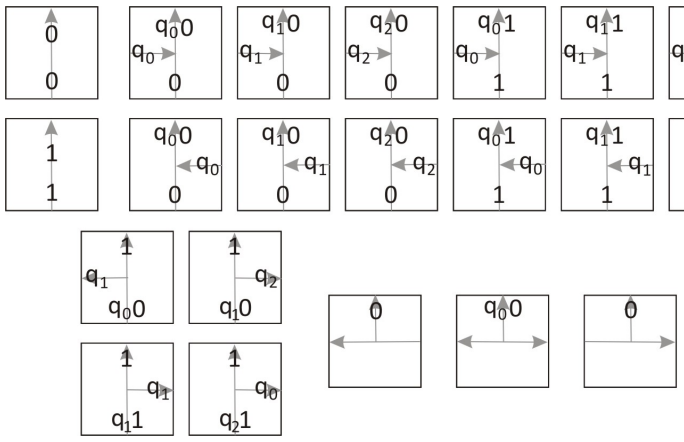


За сваки симбол траке s_k уводимо по једну плочицу, а за сваки пар $(q_i, s_k) \in Q \times \Gamma$ уводимо две *плочице везе.* За наредбе $q_i s_k s_m L q_j$ и $q_i s_k s_m R q_j$ уводимо *плочице акције.* Почетне захтеве, тј. празну траку одређују *почетне плочице* (s_0 је бланко знак, q_0 је почетно стање).

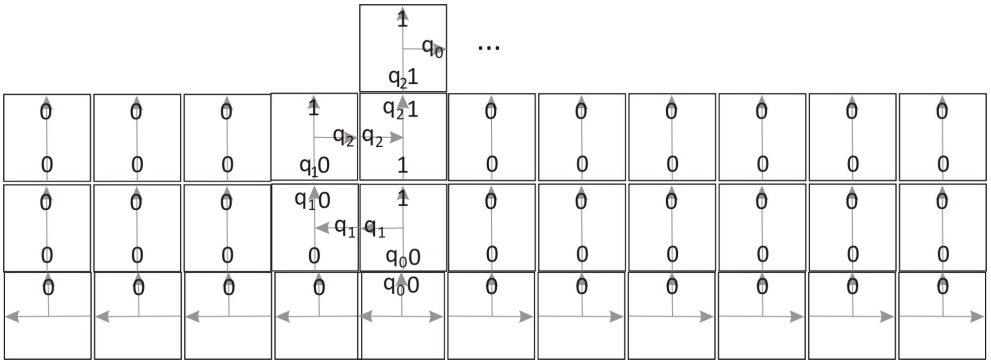
Пример 61. За симулацију Тјурингове машине:

$$q_0 01 L q_1,\ q_1 01 R q_2,\ q_1 11 R q_1,\ q_2 11 R q_0$$

користимо следеће плочице, подразумевајући да је 0 бланко знак.



Израчунавање дате машине на празној траци генерише одговарајуће ређање уведених плочица



Ако $\langle T \rangle$ означава скуп плочица одређених Тјуринговом машином T, онда није тешко уочити да важи: Тјурингова машина T се не зауставља на празној траци акко плочицама из $\langle T \rangle$ је могуће прекрити полураван са одговарајућом првом врстом.

ВРЕМЕНСКЕ КЛАСЕ СЛОЖЕНОСТИ **P** И **NP**

Главни задатак у теорији сложености јесте класификација рекурзивних функција и рекурзивних скупова (језика, проблема). Класе сложености углавном се дефинишу уз ограничење ресурса (пре свега времена, одн. простора) којима располаже механизам који извршава алгоритам и анализирањем „најгорег случаја“.

Због једноставности, ограничићемо се само на рекурзивне функције, одн. скупове над алфабетом $\Sigma = \{0,1\}$. Наравно, ово не представља никакво суштинско ограничење, будући да се сви коначни објекти могу кодирати речима из $\{0,1\}$. Подсећамо:

Језик $L \subseteq \Sigma^*$ је рекурзиван акко је функција $\chi_L : \Sigma^* \to \{0,1\}$,

$$\chi_L(w) = \begin{cases} 1, & w \in L, \\ 0, & w \notin L. \end{cases}$$

рекурзивна.

- функција $f : \Sigma^* \to \Sigma^*$ је рекурзивна ако постоји Тјурингова машина $(Q, \mathsf{q}_0, \{\mathsf{q}_{\text{stop}}\}, \{0,1\}, \sqcup, \Gamma, \tau)$ таква да за све $w \in \Sigma^*$,

$$\mathsf{q}_0 w \to^* \mathsf{q}_{\text{stop}} f(w);$$

- скуп (језик) $L \subseteq \Sigma^*$ је рекурзиван ако постоји Тјурингова машина $\mathbb{T} = (Q, \mathsf{q}_0, \{\mathsf{q}_{\text{da}}, \mathsf{q}_{\text{ne}}\}, \{0,1\}, \sqcup, \Gamma, \tau)$ таква да за све $w \in \Sigma^*$,

$$\mathsf{q}_0 w \to^* \begin{cases} \mathsf{q}_{\text{da}} \sqcup, & w \in L, \\ \mathsf{q}_{\text{ne}} \sqcup, & w \notin L. \end{cases}$$

Нека је $\mathbb{T} = (Q, \mathsf{q}_0, F, \{0,1\}, \sqcup, \Gamma, \tau)$ нека Тјурингова машина која се зауставља за сваку реч улазног алфабета: за свако $w \in \Sigma$, $\mathbb{T}$ генерише коначно израчунавање

$$\mathsf{q}_0 w \to \mathrm{C}_1 \to \mathrm{C}_2 \to \cdots \to \mathrm{C}_{\text{stop}} \equiv u\mathsf{q}v \in \Gamma^* F \Gamma^+,$$

чију дужину означавамо $\text{time}_{\mathbb{T}}(w)$.

Дефиниција 19. *Тјурингова машина $\mathbb{T}$, која се зауставља за сваку реч улазног алфабета, ради у времену $t : \mathbb{N} \to \mathbb{N}$ ако за свако $w \in \Sigma^*$ важи $\text{time}_{\mathbb{T}}(w) \leqslant t(|w|)$ (тј. израчунавање за улаз w се завршава за $\leqslant t(|w|)$ корака).*

Дефиниција 20. *Нека $t : \mathbb{N} \to \mathbb{N}$.*

1. *Рекурзивна функција $f : \Sigma^* \to \Sigma^*$ је израчунљива у времену t, ако постоји ТМ која је израчунава и ради у времену $\mathrm{O}(t)$.*
2. *Језик $L \subseteq \Sigma^*$ је одлучив у времену t, ако постоји ТМ која га одлучује и ради у времену $\mathrm{O}(t)$.*

Класе сложености се дефинишу за тзв. временски конструктибилне функције.

Дефиниција 21. *1) Функција $t : \mathbb{N} \to \mathbb{N}$ је временски конструктибилна ако је $t(n) \geqslant n$, за све n и функција $w \mapsto \text{bin}(t(|w|))$ је израчунљива у времену t.*

$\text{bin}(t(|w|))$ је бинарна репрезентација броја $t(|w|)$

2) Класа временске сложености одређена функцијом $t : \mathbb{N} \to \mathbb{N}$, *у ознаци* $\mathrm{TIME}(t)$, *јесте скуп свих језика над* $\{0,1\}$ *који су одлучиви у времену* t.

Примери временски конструктибилних функција су $n \mapsto n^2$, $n \mapsto n^3$, $n \mapsto 2^n$ итд. Уопште, сваки полином је временски конструктибилна функција.

Нека је $\mathbb{N}[x]$ скуп свих полинома са једном променљивом x и коефицијентима из $\mathbb{N}$. Класа језика (проблема) која заузима значајно место у теорији сложености јесте

$$\mathbf{P} \stackrel{\text{def}}{=} \bigcup_{P(x)\in\mathbb{N}[x]} \mathrm{TIME}(P).$$

Дакле, $L \in \mathbf{P}$ акко постоји полином $P(x) \in \mathbb{N}[x]$ и постоји Тјурингова машина $\mathbb{T}$ која одлучује L и ради у времену P.

Пример 62. $\mathsf{PAL} = \{w \in \{0,1\}^* \mid w \text{ је палиндром}\} \in \mathbf{P}$.

Теорема 33. *Нека је* $L_1, L_2 \subseteq \Sigma^*$ *и* $L_1, L_2 \in \mathbf{P}$. *Тада* $L_1^{\complement}, L_1 \cap L_2, L_1 \cup L_2 \in \mathbf{P}$.

Доказ. Нека је $\mathbb{T}_i$ Тјурингова машина која ради у времену $P_i(n)$ и прихвата језик L_i, $i = 1, 2$.

Доказ да $L_1^{\complement} \in \mathbf{P}$ је једноставан: довољно је покренути машину $\mathbb{T}_1$ за улаз w, чекати да се она заустави и дати супротан одговор од $\mathbb{T}_1$. Ова нова машина очигедно ради у полиномијалном времену.

Конструкцију Тјурингове машине која у полиномијалном времену одлучује $L_1 \cap L_2$, одн. $L_1 \cup L_2$, изостављамо, претпостављајући да је интуитивно јасно да, у оба случаја, таква машина постоји. □

Дефиниција 22. *Функција* $f : \Sigma^* \to \Sigma^*$ *је израчунљива у полиномијалном времену ако постоји Тјурингова машина* $\mathbb{T}$ *која израчунава* f *за време* P, *где је* $P(x)$ *неки полином из* $\mathbb{N}[x]$.

Дефиниција 23. *Нека је* $L_1, L_2 \subseteq \Sigma^*$. *Проблем* L_1 *се своди у полиномијалном времену на проблем* L_2, *у ознаци* $L_1 \leqslant_{\mathrm{P}} L_2$, *ако постоји функција* $f : \Sigma^* \to \Sigma^*$ *израчунљива у полиномијалном времену таква да за све* $w \in \Sigma^*$

$$w \in L_1 \Leftrightarrow f(w) \in L_2$$

(при чему користимо ознаку $f : L_1 \leqslant_{\mathrm{P}} L_2$*).*

Теорема 34. *Релација* $\leqslant_{\mathrm{P}}$ *је рефлексивна и транзитивна.*

Доказ. Рефлексивност је очигледна јер је $L \leqslant_{\mathrm{P}} L$, за сваки проблем L, будући да идентичко пресликавање $\mathrm{id} : \Sigma^* \to \Sigma^*$ испуњава тражене услове, $\mathrm{id} : L \leqslant_{\mathrm{P}} L$.

Да бисмо доказали транзитивност, претпоставимо да $L_i \subseteq \Sigma^*$, $i = 1,2,3$ и $f : L_1 \leqslant_{\mathrm{P}} L_2$, $g : L_2 \leqslant_{\mathrm{P}} L_3$, при чему су f и g израчунљиве функције редом у полиномијалним временима P_f и P_g на Тјуринговим машинама $\mathbb{T}_f$ и $\mathbb{T}_g$. Тада за све $w \in \Sigma^*$,

$$w \in L_1 \Leftrightarrow f(w) \in L_2 \Leftrightarrow g(f(w)) \in L_3,$$

при чему је композиција $g \circ f$ израчунљива у полиномијалном времену $n \mapsto P_f(n) + P_g(n + P_f(n))$. Прецизније, рад машине $\mathbb{T}_f$ за улаз $w \in \Sigma^*$ даје излаз $f(w)$, при чему важи:

$$|f(w)| \leqslant |w| + |T_f(|w|)|,$$

јер је немогуће да $\mathbb{T}_f$ произведе излаз дужи од збира дужине улаза $|w|$ и броја корака који су јој потребни. Машина $\mathbb{T}_g$ даље ради за улаз $f(w)$, па је време које је потребно машини $\mathbb{T}_g$ за улаз $f(w)$ није дуже од $P_g(|f(w)|) \leqslant P_g(|w| + P_f(|w|))$. □

Теорема 35. *Ако је* $L_1 \leqslant_{\mathrm{P}} L_2$ *и* $L_2 \in \mathbf{P}$, *онда* $L_1 \in \mathbf{P}$.

Доказ. Претпоставимо да $f : L_1 \leqslant_{\mathrm{P}} L_2$, при чему је f израчунљиво у полиномијалном времену P_f. Нека је $\mathbb{T}_2$ недетерминистичка Тјурингова машина која прихвата L_2 у полиномијалном времену P. Конструишимо Тјурингову машину $\mathbb{T}_1$ на следећи начин: за улаз $w \in \Sigma^*$

- прво се одређује $f(w)$ – за ово је потребно $P_f(|w|)$ корака, при чему се добија излаз дужине $\leqslant |w| + P_f(|w|)$,
- затим се „позива“ $\mathbb{T}_2$ за улаз $f(w)$ и даје исти одговор као $\mathbb{T}_2$ – потребан број корака је $P(|f(w)|) \leqslant P(|w| + P_f(|w|))$, што је полиномијално ограничење.

Дакле, $\mathbb{T}_1$ ради у полиномијалном времену. Такође,

$$w \in L_1 \Leftrightarrow f(w) \in L_2 \Leftrightarrow \mathbb{T}_2 \text{ прихвата } f(w),$$

што значи да $\mathbb{T}_1$ прихвата језик L_1. □

Пример 63. За алфабет исказне логике можемо узети скуп

$$\{p, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\},$$

при чему речи $p, pp, ppp, pppp, \ldots$ користимо као исказна слова $p_1, p_2, p_3, p_4, \ldots$ Дакле, формули $(p_2 \Rightarrow (\neg p_3 \wedge p_1))$ одговара реч $(pp \Rightarrow (\neg ppp \wedge p))$ поменутог алфабета. Наравно, исказне формуле можемо представљати и речима алфабета $\{0,1\}$ користећи, на пример, следеће кодирање:

$$\begin{pmatrix} p & \neg & \wedge & \vee & \Rightarrow & \Leftrightarrow & (&) \\ 000 & 001 & 010 & 100 & 011 & 101 & 110 & 111 \end{pmatrix}.$$

Тада је кôд формуле $(pp \Rightarrow (\neg ppp \wedge p))$ једнак:

$$11000000001111000100000000010000111111.$$

Једноставно се уочава да

$$F = \{w \in \{0,1\}^* \mid w \text{ је кôд неке исказне формуле}\} \in \mathbf{P}.$$

Приметимо да ако $w \in F$, онда се у одговарајућој формули, коју ћемо означити φ_w могу појављивати само нека од слова $p_1, p_2, \ldots, p_{|w|}$. За сваку валуацију $v : \{p_1, \ldots, p_{|w|}\} \to \{0,1\}$ потпуно је одређена истинитосна вредност $\varphi_w[v]$. Валуацију v можемо идентификовати са $\{0,1\}$-низом дужине $|w|$. Ако постоји валуација v таква да је $\varphi_w[v] = 1$, односно $v \models \varphi_w$, кажемо да је φ_w задовољива формула. Нека је

$$\mathrm{SAT} = \{w \in \{0,1\}^* \mid \varphi_w \text{ је задовољива формула}\}.$$

Дакле, $w \in \mathrm{SAT} \Leftrightarrow \exists v \in \{0,1\}^{|w|}\ v \models \varphi_w$. Није тешко уочити да је „$v \models \varphi_w$“ одлучиво у полиномијалном времену.

Дефиниција 24. *Класа* **NP** *садржи све језике* $L \subseteq \{0,1\}^*$ *за које постоји полином* $q(x) \in \mathbb{N}[x]$ *и релација* $R \subseteq \{0,1\}^* \times \{0,1\}^*$ *из* **P** *такви да за све* $w \in \{0,1\}^*$ *важи:*

$$w \in L \Leftrightarrow \exists v \in \{0,1\}^{q(|w|)}\ R(w,v).$$

Релација R *се назива верификатор за* L. *Реч* $v \in \{0,1\}^{q(|w|)}$ *таква да* $R(w,v)$ *назива се сведок да* $w \in L$.

Пример 64. $\mathrm{SAT} \in \mathbf{NP}$.

Једна од најчешћих карактеризација класе **NP** јесте она помоћу тзв. недетерминистичких Тјурингових машина. Недетерминистичка Тјурингова машина јесте уређена седморка $(Q, \mathsf{q}_0, F, \Sigma, \sqcup, \Gamma, \tau)$, где је $\tau : (Q \setminus F) \times \Gamma \to \mathcal{P}(\Gamma \times \{\mathsf{L}, \mathsf{P}, \mathsf{R}\} \times Q)$. Ако је

$$\tau(\mathsf{q}, \mathsf{s}) = \{\mathsf{s}_1\mathsf{R}\mathsf{q}_1, \mathsf{s}_2\mathsf{L}\mathsf{q}_2, \ldots, \mathsf{s}_k\mathsf{P}\mathsf{q}_k\},$$

онда конфигурација $\cdots \mathsf{s}_L\mathsf{q}\mathsf{s}\mathsf{s}_D \cdots$ има k наследника:

$$\cdots \mathsf{s}_L\mathsf{s}_1\mathsf{s}_D\mathsf{q}_1 \cdots ;\ \cdots \mathsf{q}_2\mathsf{s}_L\mathsf{s}_2\mathsf{s}_D \cdots ;\ \ldots ;\ \cdots \mathsf{s}_L\mathsf{q}_k\mathsf{s}_k\mathsf{s}_D \cdots$$

Свака НТМ $\mathbb{T}$ за улаз $w \in \Sigma^*$ генерише једно дрво, чији је корен почетна конфигурација одређена као у случају обичних (детерминистичких) Тјурингових машина. Свака грана дрвета представља једно израчунавање машине $\mathbb{T}$ за улаз w. У наставку ћемо посматрати недетерминистичке Тјурингове машине са два завршна стања, $F = \{\mathsf{q}_{\mathrm{da}}, \mathsf{q}_{\mathrm{ne}}\}$, такве да су за сваку реч $w \in \Sigma^*$ сва израчунавања коначна. Завршне конфигурације у стању q_{da} називамо конфигурацијама прихватања, а завршне конфигурације у стању q_{ne} конфигурацијама одбацивања.

Дефиниција 25. *НТМ* $\mathbb{T}$ *одлучује језик* $L \subseteq \Sigma^*$ *ако за свако* $w \in \Sigma^*$ *важи:* $w \in L$ *акко постоји израчунавање машине* $\mathbb{T}$ *за улаз* w *које се завршава конфигурацијом прихватања.*

$w \notin L$ акко се сва израчунавања машине $\mathbb{T}$ за улаз w завршавају конфигурацијом одбацивања.

Дефиниција 26. *1) НТМ* $\mathbb{T}$ *ради у времену* $t : \mathbb{N} \to \mathbb{N}$ *ако за све* $w \in \Sigma^*$ *свако израчунавање машине* $\mathbb{T}$ *за улаз* w *није дуже од* $t(|w|)$.

2) $\mathrm{NTIME}(t)$ *је скуп свих језика над* $\{0,1\}$ *које одлучује нека НТМ која ради у времену* $\mathrm{O}(t)$.

Теорема 36. $\mathbf{NP} = \bigcup_{k \geqslant 1} \mathrm{NTIME}(n^k)$

Скица доказа. ($\subseteq$) Претпоставимо да $L \in \mathbf{NP}$. Нека је p полином и R верификатор језика L:

$$w \in L \text{ акко } \exists v \in \{0,1\}^{p(|w|)}\, R(w,v).$$

Треба конструисати НТМ $\mathbb{T}$ која одлучује L у полиномијалном времену. Нека је $\mathbb{T}_R$ обична Тјурингова машина која одлучује R у времену $q(x) \in \mathbb{N}[x]$.

$\mathbb{T}$: за улаз w

1. „погоди“ реч v дужине $p(|w|)$;
   (Сваки полином је временски конструктибилна функција.)
2. симулирај $\mathbb{T}_R$ за улаз (w,v);
3. врати одговор „ДА“ ако $\mathbb{T}_R(w,v) \downarrow \mathsf{q}_{\mathrm{da}}$ и
   врати одговор „НЕ“ ако $\mathbb{T}_R(w,v) \downarrow \mathsf{q}_{\mathrm{ne}}$.

($\supseteq$) Претпоставимо да L одлучује НТМ $\mathbb{T}$ која ради у полиномијалном времену $p(x) \in \mathbb{N}[x]$. Без губљења општости можемо претпоставити да сви скупови $\tau(\mathsf{q},\mathsf{s})$, $\mathsf{q} \in Q \setminus F$, $\mathsf{s} \in \Gamma$, имају исти број елемената. Нека је $|\tau(\mathsf{q},\mathsf{s})| = k$, за све $\mathsf{q} \in Q \setminus \{\mathsf{q}_{\mathrm{da}}, \mathsf{q}_{\mathrm{ne}}\}$ и $\mathsf{s} \in \Gamma$. Претпоставимо и да су скупови $\tau(\mathsf{q},\mathsf{s})$ уређени. Уз ове претпоставке, сваки унутрашњи чвор дрвета, које генерише $\mathbb{T}$ за улаз w, има тачно k потомака, који су на неки начин уређени. Тада је сваком завршном чвору Z дрвета придружена једна реч над алфабетом $\{1,2,\ldots,k\}$ дужине $p(|w|)$. Ове речи можемо схватити као записе природних бројева у бази k и превести их у бинарни запис, одакле следи да постоји полином $p_1(x) \in \mathbb{N}[x]$ такав да сваком завршном чвору Z, дрвета израчунавања за улаз w, одговара реч $v_Z \in \{0,1\}^{p_1(|w|)}$. Реч v_Z заправо представља израчунавање које одговара грани која се завршава чвором Z.

Ако немају можемо додати инструкције без неког посебног ефекта; нпр. $\mathsf{qssP}\mathsf{q}_{\mathrm{ne}}$.

Конструишимо обичну Тјурингову машину $\mathbb{T}_R$:

$\mathbb{T}_R$: за улаз (w, v)

1. одреди завршну конфигурацију израчунавања машине $\mathbb{T}$ за улаз w ако је кôд тог израчунавања реч v;
2. ако је завршна конфигурација нека конфигурација прихватања врати „ДА“, а ако је завршна конфигурација нека конфигурација одбацивања врати „НЕ“.

$\mathbb{T}_R$ ради у полиномијалном времену и дефинише верификатор R језика L:

$$w \in L \text{ акко } \exists v \in \{0,1\}^{p_1(|w|)}\ R(w, v).$$

□

Теорема 37. *1)* $\mathbf{P} \subseteq \mathbf{NP}$

2) **NP** *је затворено за унију и пресек: ако* $L_1, L_2 \in \mathbf{NP}$, *онда* $L_1 \cap L_2, L_1 \cup L_2 \in \mathbf{NP}$.

Отворен проблем. $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

Дефиниција 27. $L \in \mathbf{coNP}$ *акко* $L^{\complement} \in \mathbf{NP}$

Отворен проблем. $\mathbf{NP} \stackrel{?}{=} \mathbf{coNP}$

Теорема 38. *1.* $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$

2. Ако $\mathbf{NP} \subseteq \mathbf{coNP}$, *онда* $\mathbf{NP} = \mathbf{coNP}$.

3. Ако $\mathbf{coNP} \subseteq \mathbf{NP}$, *онда* $\mathbf{NP} = \mathbf{coNP}$.

4. Ако $L_1 \leq_{\mathrm{P}} L_2$ *и* $L_2 \in \mathbf{NP}$, *онда* $L_1 \in \mathbf{NP}$.

5. Ако $L_1 \leq_{\mathrm{P}} L_2$ *и* $L_2 \in \mathbf{coNP}$, *онда* $L_1 \in \mathbf{coNP}$.

Доказ. 1) $\mathbf{P} \subseteq \mathbf{NP}$ и **P** је затворено за комплементе.
2) Нека је $\mathbf{NP} \subseteq \mathbf{coNP}$. Ако $L \in \mathbf{coNP}$, онда $L^{\complement} \in \mathbf{NP} \subseteq \mathbf{coNP}$.
Дакле, $L = (L^{\complement})^{\complement} \in \mathbf{NP}$. □

NP-КОМПЛЕТНОСТ

Дефиниција 28. *Проблем L је* **NP***-комплетан ако* $L \in \mathbf{NP}$ *и за свако* $L' \in \mathbf{NP}$ *важи* $L' \leq_{\mathrm{P}} L$.

Аналогно се дефинише појам **coNP**-комплетности. (Може се доказати да је проблем L **NP**-комплетан акко је L^{C} **coNP**-комплетан.)

Теорема 39. *Ако је* L_1 **NP***-комплетан,* $L_2 \in \mathbf{NP}$ *и* $L_1 \leq_{\mathrm{P}} L_2$, *онда је* L_2 **NP***-комплетан.*

Доказ. Нека је $L' \in \mathbf{NP}$ произвољан проблем. Како је L_1 **NP**-комплетан, закључујемо да је $L' \leq_{\mathrm{P}} L_1$. Из $L_1 \leq_{\mathrm{P}} L_2$, на основу транзитивности релације $\leq_{\mathrm{P}}$, закључујемо да $L' \leq_{\mathrm{P}} L_2$. Будући да $L_2 \in \mathbf{NP}$, следи да је L_2 **NP**-комплетан. □

Теорема 40. *Нека је* L **NP***-комплетан.*

1. *Ако* $L \in \mathbf{P}$, *онда* $\mathbf{P} = \mathbf{NP}$.

2. *Ако* $L \in \mathbf{coNP}$, *онда* $\mathbf{NP} = \mathbf{coNP}$.

Доказ. 1. За произвољан $L' \in \mathbf{NP}$ важи $L' \leq_{\mathrm{P}} L$, па ако $L \in \mathbf{P}$, онда $L' \in \mathbf{P}$. Дакле, $\mathbf{NP} \subseteq \mathbf{P}$, па тиме и $\mathbf{P} = \mathbf{NP}$ (јер је $\mathbf{P} \subseteq \mathbf{NP}$).

2. Претпоставимо да $L \in \mathbf{coNP}$. Нека је $L' \in \mathbf{NP}$ произвољан језик. Како је L **NP**-комплетан, следи да је $L' \leqslant_{\mathrm{P}} L$, па $L' \in \mathbf{coNP}$. Дакле, $\mathbf{NP} \subseteq \mathbf{coNP}$, а тиме је и $\mathbf{NP} = \mathbf{coNP}$. □

Теорема 41. [*Кук-Левинова теорема*] SAT *је* **NP***-комплетан.*

Доказ. Већ је истакнуто да SAT $\in$ NP.

Нека је L произвољан језик из **NP** и

$$\mathbb{T} = (Q, \mathsf{q}_0, \{\mathsf{q}_{\mathrm{da}}, \mathsf{q}_{\mathrm{ne}}\}, \Gamma, \sqcup, \Sigma, \delta)$$

НТМ која одлучује L и ради у времену $p(n)$, где је p неки полином. За свако $w \in \Sigma^*$, конструисаћемо у полиномијалном времену једну исказну формулу $\varphi_{\mathbb{T},w}$ такву да важи:

$$\mathbb{T}(w) \downarrow q_{\mathrm{da}} \text{ акко } \varphi_{\mathbb{T},w} \in \mathrm{SAT};$$

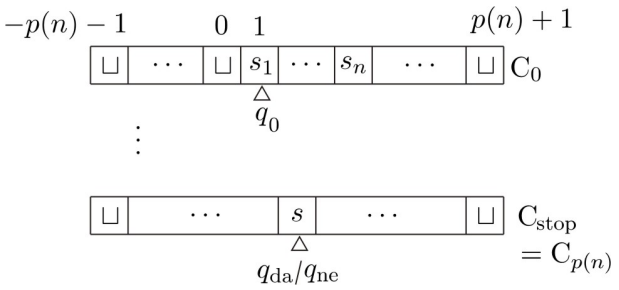
тиме доказујемо да $L \leq_{\mathrm{P}} \mathrm{SAT}$.

Да бисмо то постигли, приметимо да за сваки улаз $w = s_1 \cdots s_n \in \Sigma^*$ дужине n:

- до краја израчунавања глава машине $\mathbb{T}$ не напушта регион који чине $2p(n)+2$ суседних ћелија траке; доделимо овим ћелијама целобројне адресе

$$-p(n)-1,\ldots,0,1,\ldots,p(n)+1$$

као на слици десно, при чему се улаз дужине n уноси у поља чије су адресе $1,\ldots,n$;



- свако израчунавање машине $\mathbb{T}$ тривијално се може продужити до израчунавања дужине $p(n)$ понављањем завршне конфигурације довољан број пута.

$$\overbrace{\mathrm{C}_0^w \to_{\mathbb{M}} \cdots \to_{\mathbb{M}} \mathrm{C}_{\mathrm{stop}}}^{\leq p(n)} \to_{\mathbb{M}} \mathrm{C}_{\mathrm{stop}} \cdots \to_{\mathbb{M}} \mathrm{C}_{\mathrm{stop}}$$
$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{=p(n)}$$

Формулу $\varphi_{\mathbb{T},w}$ састављамо користећи исказна слова q_a^i, s_a^i, за $q \in Q$, $s \in \Gamma$, $0 \leqslant i \leqslant p(n)$ и $-p(n)-1 < a < p(n)+1$. Слово q_a^i одговара исказу „глава у стању q скенира ћелију чија је адреса a у i-тој конфигурацији израчунавања“. Слово s_a^i одговара исказу „симбол s уписан је у ћелију са адресом a у i-тој конфигурацији израчунавања“. Будући да слова чији је горњи индекс i користимо за описивање i-те конфигурације израчунавања, формула

$$\sigma_i \equiv \bigwedge_a \bigvee_s s_a^i \wedge \bigwedge_a \bigwedge_{s \neq s'} \neg(s_a^i \wedge s_a'^i) \wedge \bigvee_a \bigvee_q \left(q_a^i \wedge \bigwedge_{a' \neq a} \bigwedge_q \neg q_{a'}^i \right),$$

значи да је у i-тој конфигурацији у сваку ћелију уписан тачно један симбол из Γ и да тачно једну ћелију скенира глава у неком стању.

Да је коректно формирана почетна конфигурација за улазну реч $w = s_1 \cdots s_n$ описујемо формулом: $\sigma_w \equiv q_{01}^0 \wedge \bigwedge_{i=1}^{n} s_{ii}^0 \wedge \bigwedge_{a \notin \{1,\ldots,n\}} \sqcup_a^0$.

Промене конфигурација током израчунавања машине $\mathbb{T}$ описујемо формулом $\sigma_{\mathbb{T}}$ која је конјункција следећих формула: за све $q \in Q \setminus \{q_{\mathrm{da}}, q_{\mathrm{ne}}\}$ и $s \in \Gamma$,

$$q_a^i \wedge s_a^i \Rightarrow \bigvee_{(q,s,s',\mathrm{R},q') \in \delta} \left(s_a'^{i+1} \wedge q_{a+1}'^{i+1} \right), \qquad q_a^i \wedge s_a^i \Rightarrow \bigvee_{(q,s,s',\mathrm{P},q') \in \delta} \left(s_a'^{i+1} \wedge q_a'^{i+1} \right),$$

$$q_a^i \wedge s_a^i \Rightarrow \bigvee_{(q,s,s',\mathrm{L},q') \in \delta} \left(s_a'^{i+1} \wedge q_{a-1}'^{i+1} \right), \qquad s_a^i \wedge \bigwedge_q \neg q_a^i \Rightarrow s_a^{i+1}, \qquad \bigwedge_a \left(q_{\mathrm{da}\,a}^i \wedge s_a^i \Rightarrow q_{\mathrm{da}\,a}^{i+1} \wedge s_a^{i+1} \right).$$

Најзад, да се израчунавање завршава конфигурацијом прихватања описује формула: $\sigma_{\mathrm{da}} \equiv \bigvee_a q_{\mathrm{da}\,a}^{p(n)}$.

Жељена формула $\varphi_{\mathbb{T},w}$ јесте формула $\sigma_w \wedge \bigwedge_i \sigma_i \wedge \sigma_{\mathbb{M}} \wedge \sigma_{\mathrm{da}}$.

НАПОМЕНА. Није тешко уочити да се $\varphi_{\mathbb{T},w}$ у полиномијалном времену може превести у *KNF*, одакле следи да је и KNF-SAT NP-комплетан. □

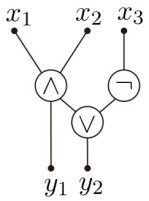
Сложеност Булових кола

Булово коло је ацикличан граф у коме разликујемо три врсте чворова:

- *улазне чворове* (на које не показује ниједна стрелица; in degree $=$ 0);
- *пролазе* који могу бити AND (in degree $= 2$), OR (in degree $= 2$) или NOT (in degree $= 1$);
- *излазне чворове* (in degree $= 1$, out degree $= 0$).

Улазни чворови могу добити вредност 0 или 1. Пролази израчунавају Булове функције $\wedge$, $\vee$ и $\neg$. Када су дате вредности улазних чворова, онда су на јединствен начин одређене вредности излазних чворова. Коло $\mathbb{C}$ са n улазних чворова $x_1, \ldots, x_n$ и m излазних чворова $y_1, \ldots, y_m$ одређује јединствену функцију $f_{\mathbb{C}} : \{0,1\}^n \to \{0,1\}^m$.

Пример 65. На слици десно приказано је Булово коло са три улазна и два излазна чвора. Ако је $x_1 = 0$, $x_2 = 1$ и $x_3 = 0$, онда је $y_1 = 0$ и $y_2 = 1$.



Два кола $\mathbb{C}$ и $\mathbb{C}'$ су еквивалентна ако одређују исту функцију, $f_{\mathbb{C}} = f_{\mathbb{C}'}$. У наставку ћемо углавном посматрати кола која имају само један излазни чвор, тј. која одређују тзв. *Булове везнике* $f : \{0,1\}^n \to \{0,1\}$.

Задатак. За неколико конкретних вредности n, конструисати коло које одређује функцију $\text{parity}_n : \{0,1\}^n \to \{0,1\}$,

$\text{parity}_n(x_1, \ldots, x_n) = 1 \Leftrightarrow 1$ се појављује непаран број пута међу улазним аргументима.

Булова кола користимо да бисмо тестирали припадање језицима алфабета $\{0,1\}$. Будући да свако коло „очекује“ улазе фиксиране дужине, а језик може садржавати речи различитих дужина, за тестирање припадања језику користимо низ кола $(\mathbb{C}_n)$ – по једно коло за сваку дужину улаза:

$$L = L \cap \Sigma^* = \underbrace{(L \cap \Sigma^0)}_{\mathbb{C}_0} \cup \underbrace{(L \cap \Sigma^1)}_{\mathbb{C}_1} \cup \cdots \cup \underbrace{(L \cap \Sigma^n)}_{\mathbb{C}_n} \cup \cdots$$

Наравно, не мора бити никакве униформности у низу кола $(\mathbb{C}_n)$; коло $\mathbb{C}_n$ ради само са улазима дужине n и не мора имати никакве сличности са осталим колима.

Дефиниција 29. *Низ кола* $(\mathbb{C}_n) = (\mathbb{C}_0, \mathbb{C}_1, \ldots, \mathbb{C}_n, \ldots)$, *при чему* $\mathbb{C}_n$ *има* n *улазних чворова и један излазни чвор, одлучује језик* $L \subseteq \{0,1\}^*$ *ако за свако* $w \in \{0,1\}^*$ *важи:*

$$w \in L \Leftrightarrow f_{\mathbb{C}_{|w|}}(w) = 1.$$

Величина кола $\mathbb{C}$ је број AND и OR пролаза које то коло садржи, и обележава се $|\mathbb{C}|$. Другим речима, $|\mathbb{C}|$ представља број излазних ивица из AND и OR пролаза.

Број NOT пролаза не утиче на величину кола.

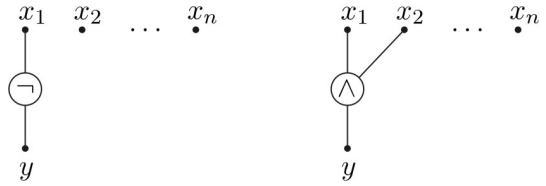
Коло је минималне величине ако не постоји мање еквивалентно коло. Проблем минимизације кола има очигледне практичне примене, али је у принципу „доста тежак“.

Дефиниција 30. *Нека* $t : \mathbb{N} \to \mathbb{N}$. *Низ кола* $(\mathbb{C}_n)$ *је величине* t *ако је* $|\mathbb{C}_n| \leqslant t(n)$, $n \in \mathbb{N}$.

SIZE(t) *је скуп свих језика* $L \subseteq \{0,1\}^*$ *које одлучује неки низ кола* $(\mathbb{C}_n)$ *величине* $\mathrm{O}(t)$.

ПРИМЕР 66. Због „неуниформности“ низ кола једноставно се уочава да чак SIZE(1) садржи неодлучиве језике. За функцију $g : \{2,3,4,\ldots\} \to \{0,1\}$, конструишимо низ кола $(\mathbb{C}_n)_{n\geqslant 2}$ на следећи начин:

ако је $g(n) = 0$; ако је $g(n) = 1$.



Језик L_g који одлучују кола $(\mathbb{C}_2, \mathbb{C}_3, \ldots)$ припада SIZE(1) и важи:

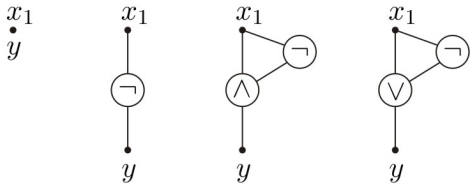
$$1^n \in L_g \Leftrightarrow g(n) = 1.$$

Одавде следи да је $g \mapsto L_g$ 1-1 функција, што значи да језика L_g има непребројиво много.

ПРИМЕР 67. Докажимо да за свако $L \subseteq \{0,1\}^*$, $L \in$ SIZE(2^n). Показаћемо, индукцијом по n, да за свако $f : \{0,1\}^n \to \{0,1\}$ постоји коло $\mathbb{C}_f$ такво да је $|\mathbb{C}_f| \leqslant 2 \cdot 2^n - 3$ и $f = f_{\mathbb{C}_f}$.

Ако је $n = 1$, функција f је једна од следећих функција:

$x \mapsto x$ $x \mapsto \neg x$ $x \mapsto 0$ $x \mapsto 1$



Величина свакаког од наведених кола једнака је $2 \cdot 2^1 - 3 = 1$.

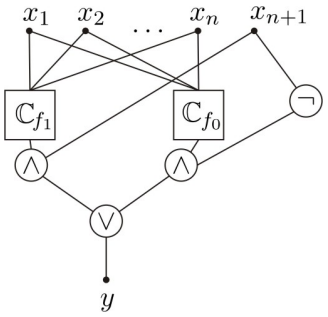
Ако је $f : \{0,1\}^{n+1} \to \{0,1\}$, нека су $\mathbb{C}_{f_0}$ и $\mathbb{C}_{f_1}$ кола која редом одређују функције:

$$f_0(x_1,\ldots,x_n) \stackrel{\text{def}}{=} f(x_1,\ldots,x_n,0) \text{ и } f_1(x_1,\ldots,x_n) \stackrel{\text{def}}{=} f(x_1,\ldots,x_n,1).$$

На основу познате једнакости

$$f(x_1,\ldots,x_n,x_{n+1}) = (f(x_1,\ldots,x_n,0) \wedge \neg x_{n+1}) \vee (f(x_1,\ldots,x_n,1) \wedge x_{n+1}),$$

конструишемо тражено коло, приказано са десне стране.



Према индуктивној претпоставци, величина овог кола није већа од $2 \cdot (2 \cdot 2^n - 3) + 3 = 2 \cdot 2^{n+1} - 3$.

Сложеност кола је у тесној вези са временском сложеношћу: за сваки језик „мале“ временске сложености, „мала“ је и сложеност кола.

Теорема 42. *Нека је* $t : \mathbb{N} \to \mathbb{N}$ *временски-конструктибилна функција. Ако* $L \in \mathrm{TIME}(t)$, *онда* $L \in \mathrm{SIZE}(t^2)$.

Доказ. Нека је $\mathbb{T}$ Тјурингова машина која одлучује L у времену t. За свако n конструисаћемо коло $\mathbb{C}_n$ које одлучује $L \cap \{0,1\}^n$:

$$f_{\mathbb{C}_{|w|}}(w) = 1 \Leftrightarrow w \in L.$$

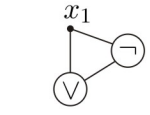
Нека је n произвољно. Израчунавања машине $\mathbb{T}$ за улазе дужине n могу се приказати у облику табеле димензија $(2t(n)+2) \times t(n)$. Слично ако у доказу Кук-Левинове теореме, довољно је посматрати само $2t(n)+2$ суседних ћелија траке, којима додељујемо целобројне адресе

$$-t(n)-1, \ldots, 0, 1, 2, \ldots, t(n)+1.$$

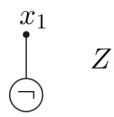
Конструисаћемо коло $\mathbb{C}_n$ са n улазних чворова $x_1, \ldots, x_n$ и једним излазним чвором y који ће добити вредност 1 ако и само ако улазни чворови редом добију вредности одређене симболима речи $w = \mathsf{s}_1 \cdots \mathsf{s}_n$ ($x_i = \mathsf{s}_i, i = 1, \ldots, n$) коју прихвата $\mathbb{T}$, тј. која припада L. Поједине значајне пролазе означићемо $Z(i, j, \mathsf{s}, \mathsf{q})$, $1 \leqslant i \leqslant t(n)$, $-t(n)-1 \leqslant j \leqslant t(n)+1$, $\mathsf{s} \in \Gamma$ и $\mathsf{q} \in Q \cup \{\varnothing\}$. Намера нам је да:

- пролаз $Z(i, j, \mathsf{s}, \mathsf{q})$ даје вредност 1 акко у i-тој конфигурацији, ћелија са адресом j садржи симбол s и скенира је глава у стању q;
- пролаз $Z(i, j, \mathsf{s}, \varnothing)$ даје вредност 1 акко у i-тој конфигурацији, ћелија са адресом j садржи симбол s и не скенира је глава.

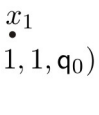
Најпре треба прецизирати пролазе који описују почетну конфигурацију.



$Z(1, j, \sqcup, \emptyset)$
$j \notin \{1, \ldots, n\}$

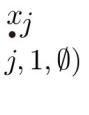


$Z(1, 1, 0, \mathsf{q}_0)$

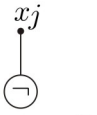


$Z(1, 1, 1, \mathsf{q}_0)$

$1 < j \leqslant n$

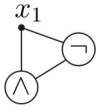


$Z(1, j, 1, \emptyset)$



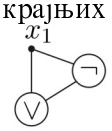
$Z(1, j, 0, \emptyset)$

Сви остали чворови $Z(1, \ldots)$ дају вредност 0.

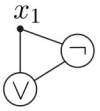


$Z(1, \ldots)$

У i-тој конфигурацији, за $1 < i \leqslant t(n)$, потпуно су одређени садржаји крајњих ћелија:



$Z(i, -t(n) - 1, \sqcup, \emptyset)$

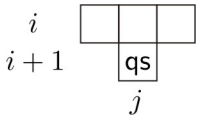


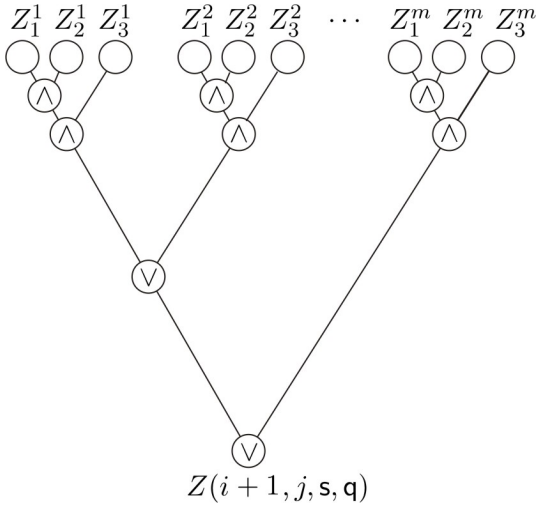
$Z(i, t(n) + 1, \sqcup, \emptyset)$

Све остали чворови $Z(\ldots, \pm(t(n)+1), \ldots)$ дају вредност 0.

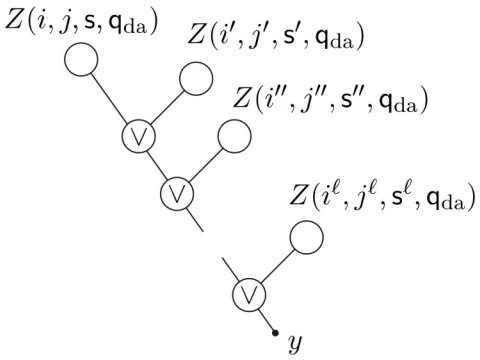
$Z(\ldots, \pm(t(n)+1), \ldots)$

За $1 \leqslant i < t(n)$, $-t(n) \leqslant j \leqslant t(n)$, $\mathsf{s} \in \Gamma$ и $\mathsf{q} \in Q \cup \{\emptyset\}$; чвор $Z(i+1, j, \mathsf{s}, \mathsf{q})$ формирамо на следећи начин. Најпре, на основу функције τ набрајамо све могуће садржаје поља $(i, j-1)$, (i, j), $(i, j+1)$ који могу довести до тога да у $(i+1)$. конфигурацији садржај ћелије j буде s и глава је скенира у стању q ако $\mathsf{q} \in Q$, одн. не скенира је ако је $\mathsf{q} = \emptyset$. Сваку од наведених могућности описује једна тројка чворова $Z(i, j-1, \cdots), Z(i, j, \cdots), Z(i, j+1, \cdots)$ која чвору $Z(i+1, j, \mathsf{s}, \mathsf{q})$ обезбеђује вредност 1. Нека су (Z_1^1, Z_2^1, Z_3^1), $(Z_1^2, Z_2^2, Z_3^2), \ldots, (Z_1^m, Z_2^m, Z_3^m)$ све тројке које обезбеђују вредност 1 чвору $Z(i+1, j, \mathsf{s}, \mathsf{q})$. Уочене чворове повезујемо као што је приказано на слици испод.





Најзад, додајемо низ OR пролаза тако да важи: ако неки чвор облика $Z(i, j, \mathsf{s}, \mathsf{q}_{\mathsf{da}})$ икада добије вредност 1, онда и чвор y добија вредност 1.



Очигледно: $f_{\mathbb{C}_{|w|}}(w) = 1 \Leftrightarrow \mathbb{T}(w) \downarrow \mathsf{q}_{\mathrm{da}}$. Пажљивом анализом конструисаног кола $\mathbb{C}_n$ закључујемо да је његова величина $\mathrm{O}(t^2)$. □

Последица 10. $\mathbf{P} \subseteq \bigcup_{k\geqslant 1} \mathrm{SIZE}(n^k) \stackrel{\mathrm{def}}{=} \mathbf{P}/\mathrm{poly}$

Није познато да ли постоји **NP**-комплетан језик који не припада $\mathbf{P}/\mathrm{poly}$. Позитиван одговор на ово питање би показао да се класе **P** и **NP** драстично разликују. Истичемо да $\mathbf{P}/\mathrm{poly}$ садржи и неодлучиве језике.

C-SAT је скраћеница од CIRCUIT-SAT

Булово коло $\mathbb{C}$, са n улазних чворова, јесте задовољиво ако постоје $x_1, \ldots, x_n \in \{0,1\}$ такви да је $f_{\mathbb{C}}(x_1, \ldots, x_n) = 1$. Подразумевајући неко кодирање Булових кола, нека је

$$\text{C-SAT} = \{[\![\mathbb{C}]\!] \mid \mathbb{C} \text{ је задовољиво Булово коло}\}.$$

Теорема 43. C-SAT *је* **NP***-комплетан.*

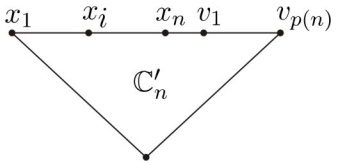
Доказ. Није тешко уочити да C-SAT $\in$ **NP**.

Нека је $L \in \mathbf{NP}$ произвољан језик и $R \subseteq \{0,1\}^* \times \{0,1\}^*$ одговарајући верификатор за L који одлучује Тјурингова машина $\mathbb{T}_R$ у времену $p(x) \in \mathbb{N}[x]$. Наравно, постоји и полином $q(x) \in \mathbb{N}[x]$ такав да за све w важи:

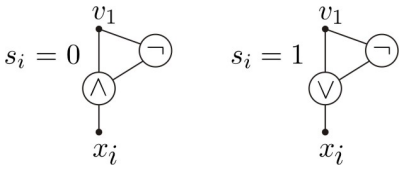
$$w \in L \Leftrightarrow \exists v \in \{0,1\}^{q(|w|)} R(w,v).$$

За све $w, v \in \{0,1\}^*$ такве да је $|w| = n$ и $|v| = q(|w|)$, израчунавање машине $\mathbb{T}_R$ за улаз (w,v) може се приказати у облику табеле димензија $(2p(|w|+q(|w|))+1)+2) \times p(|w|+q(|w|))+1)$. То значи да постоји Булово коло $\mathbb{C}'_n$ полиномијалне величине, са $n+q(n)$ улазних чворова $x_1, \ldots, x_n, v_1, \ldots, v_{q(n)}$ и једим излазним чвором y, тако да важи:

$$f_{\mathbb{C}'_n}(x_1, \ldots, x_n, v_1, \ldots, v_{q(n)}) = 1 \Leftrightarrow \mathbb{T}_R(x_1 \cdots x_n, v_1 \cdots v_{q(n)}) \downarrow \mathsf{q}_{\mathrm{da}}$$



За реч $w = s_1 \cdots s_n$, коло $\mathbb{C}_w$ формирамо од кола $\mathbb{C}'_n$ тако што на чворовима $x_1, \ldots, x_n$ фиксирамо вредности у складу са симболима речи w:



$\mathbb{C}_w$ је задовољиво $\Leftrightarrow$ постоје $v_1, \ldots, v_{q(n)} \in \{0,1\}$ т.д. $f_{\mathbb{C}_w}(v_1, \ldots, v_{q(n)}) = 1$
$\Leftrightarrow w \in L$.

На основу описане конструкције није тешко уочити да је функција $w \mapsto \mathbb{C}_w$ израчунљива у полиномијалном времену. □

3SAT проблем је специјалан случај SAT проблема. 3SAT јесте проблем одлучивања задовољивости исказних формула у конјунктивној нормалној форми са највише три литерала по клаузи.

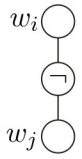
Теорема 44. 3SAT *је* **NP**-*комплетан.*

Доказ. Из досадашњих разматрања једноставно следи да 3SAT $\in$ **NP**.

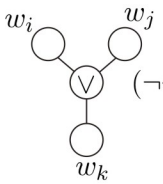
Показаћемо C-SAT $\leqslant_{\mathrm{P}}$ 3SAT. Одговарајуће свођење јесте превођење (наравно у полиномијалном времену) Буловог кола $\mathbb{C}$ у формулу $\varphi_{\mathbb{C}}$ одговарајућег облика, тако да важи:

$$\mathbb{C} \text{ је задовољиво} \quad \Leftrightarrow \quad \varphi_{\mathbb{C}} \text{ је задовољива.}$$

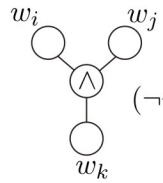
Све чворове кола $\mathbb{C}$ означимо $w_1, \ldots, w_m$, при чему је w_m излазни чвор. За сваки унутрашњи чвор (пролаз) додајемо по једну формулу:



$$(\neg w_i \Rightarrow w_j) \wedge (w_i \Rightarrow \neg w_j)$$



$$(\neg w_i \wedge \neg w_j \Rightarrow \neg w_k) \wedge (\neg w_i \wedge w_j \Rightarrow \neg w_k) \wedge (w_i \wedge \neg w_j \Rightarrow \neg w_k) \wedge (w_i \wedge w_j \Rightarrow w_k)$$



$$(\neg w_i \wedge \neg w_j \Rightarrow \neg w_k) \wedge (\neg w_i \wedge w_j \Rightarrow w_k) \wedge (w_i \wedge \neg w_j \Rightarrow w_k) \wedge (w_i \wedge w_j \Rightarrow w_k)$$

Поред наведених клауза додајемо и клаузу коју чини само исказно слово w_m (излазни чвор). Конјункција свих наведених клауза јесте тражена формула $\varphi_{\mathbb{C}}$. □

Просторна класа сложености **PSPACE**

Нека је $\mathbb{T}$ ТМ (НТМ) која се зауставља за сваки улаз. Просторна сложеност машине $\mathbb{T}$ јесте функција $S_{\mathbb{T}} : \mathbb{N} \to \mathbb{N}$ таква да је $S_{\mathbb{T}}(n)$ максималан број ћелија које скенира глава током било ког израчунавања за улазе дужине n.

$\mathrm{SPACE}(f) \stackrel{\text{def}}{=} \{L \mid L$ одлучује нека ТМ која ради у простору $O(f)\}$

$\mathrm{NSPACE}(f) \stackrel{\text{def}}{=} \{L \mid L$ одлучује нека НТМ која ради у простору $O(f)\}$

Класе просторне сложености углавном се дефинишу у односу на просторно конструктибилне функције. $S : \mathbb{N} \to \mathbb{N}$ је просторно конструктибилна ако је функција $w \mapsto \mathrm{bin}(S(|w|))$ израчунљива у простору S.

Пример 68. $\mathrm{SAT} \in \mathrm{SPACE}(n)$.

Теорема 45. [*Савичева теорема*] *Ако је* $S : \mathbb{N} \to \mathbb{N}$ *просторно-конструктибилна функција, онда је* $\mathrm{NSPACE}(S(n)) \subseteq \mathrm{SPACE}(S(n)^2)$.

Доказ. Нека је L произвољан језик из $\mathrm{NSPACE}(S(n))$ и $\mathbb{T} = (Q, q_0, \{q_{\mathrm{da}}, q_{\mathrm{ne}}\}, \Gamma, \sqcup, \Sigma, \delta)$ НТМ која одлучује L и ради у простору $S(n)$. Описаћемо како се може конструисати обична Тјурингова машина $\mathbb{T}'$ која одлучује L и ради у простору $S(n)^2$.

Приметимо најпре да за сваки улаз дужине n, до краја израчунавања глава машине $\mathbb{T}$ не напушта регион који чине $2S(n) + 2$ суседних ћелија траке; доделимо овим ћелијама целобројне адресе

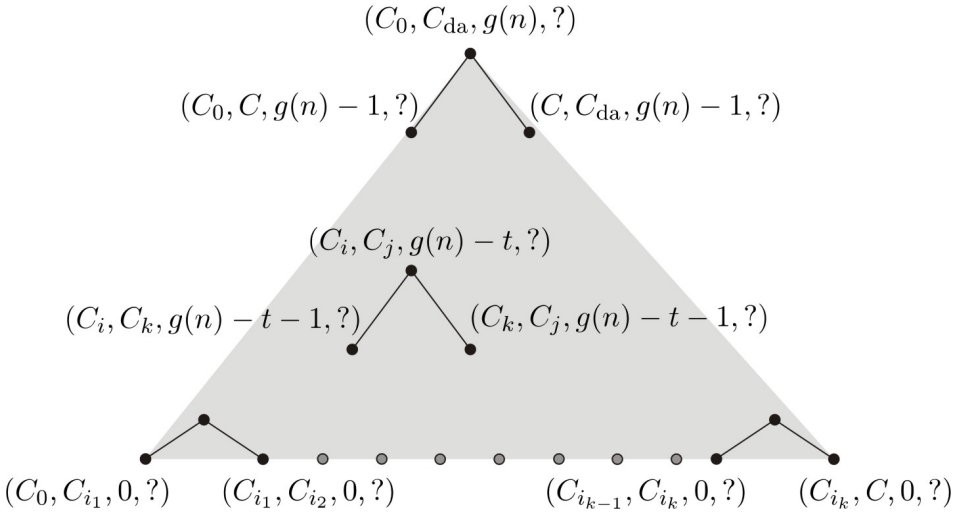
$$-S(n) - 1, \ldots, 0, 1, \ldots, S(n) + 1$$

као у доказу Кукове теореме, при чему се улаз дужине n уноси у поља чије су адресе $1, \ldots, n$.

Приметимо и да постоји константа c таква да укупан број могућих текућих конфигурација на оваквом простору није већи од $2^{cS(n)}$. Будући да се у сваком израчунавању, текуће конфигурације не понављају (јер се $\mathbb{T}$ зауставља за сваки улаз), свако израчунавање машине $\mathbb{T}$ за улазе дужине n не може бити дуже од $2^{cS(n)}$. Другим речима, машина $\mathbb{T}$ ради у времену $2^{cS(n)}$.

Ако је C_0^w почетна конфигурација за улаз w дужине n, машина $\mathbb{T}'$ треба да испита да ли се нека конфигурација прихватања C_{da} може достићи за не више од $2^{cS(n)}$ корака извршавања машине $\mathbb{T}$. Кључно запажање у овом доказу јесте: из C_0^w се може достићи C_{da} за не више од $2^{cS(n)}$ акко постоји конфигурација C таква да се из C_0^w може достићи C за не више од $2^{cS(n)-1}$ корака и из C се може достићи C_{da} за не више од $2^{cS(n)-1}$ корака. Нека $(C', C'', i, ?)$ означава питање: „Да ли се из C' може достићи C'' за не више од 2^i корака?“. Потврдан одговор на ово питање означићемо $(C', C'', i, \mathrm{da})$, а одречан $(C', C'', i, \mathrm{ne})$. Машина $\mathbb{T}'$ треба да испита

да ли постоје потребне конфигурације за које су потврдни одговори на сва питања наведена у наредном дрвету. Нека је $g(n)$ краћа ознака за $cS(n)$.



Наравно, посао машине $\mathbb{T}'$ није да формира наведено дрво, већ да (у простору $O(S(n)^2)$) испита да ли се овакво дрво може формирати. То ће постићи претраживањем листе свих могућих конфигурација. Фиксирајмо зато неко уређење свих могућих конфигурација: $C_1, C_2, \ldots, C_{f(n)}$, при чему је $f(n) \leqslant 2^{cS(n)}$. Штавише, погодним кодирањем, могуће је конструисати ТМ $\mathbb{E}$ која ради у простору $O(S(n))$ и која за задато C_i, $i < f(n)$, одређује наредну конфигурацију C_{i+1} фиксиране листе.

Уопштено говорећи, машина $\mathbb{T}'$ ће полазећи од почетног питања облика $(C_0, C, cS(n), ?)$ додавати питања са једне гране приказаног дрвета док не стигне до питања облика $(C', C'', 0, ?)$ на које се може одговорити (са da или ne) у простору константне величине (што зависи само од машине $\mathbb{T}$)...

За улаз $w \in \Sigma^n$, $\mathbb{T}'$ најпре на траци штампа $(C_0^w, C_1, g(n), ?)$. Наставак рада описујемо у општем случају. Претпоставимо да је $\mathbb{T}'$ на траци одштампала:

$$(\star)\ \ (C_0^w, C_{i_1}, g(n), ?), (C_{i_2}, C_{i_3}, g(n)-1, ?), \ldots, (C_{i_{2t-2}}, C_{i_{2t-1}}, g(n)-t+1, ?), (C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, X),$$

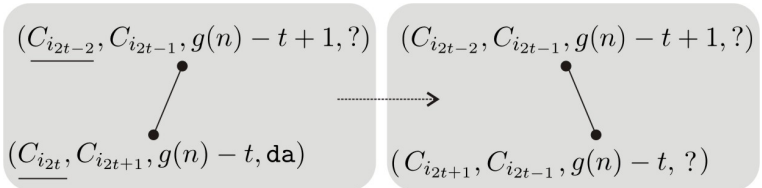
при чему је $C_{i_{2j-2}} = C_{i_{2j}}$ или $C_{i_{2j-1}} = C_{i_{2j+1}}$, $j = 1, \ldots, t$, и $X \in \{\mathrm{da}, \mathrm{ne}, ?\}$. Приметимо да је овакав садржај траке најдужи за $t = g(n) = cS(n)$ и да се у и том случају може записати у простору $O(S(n)^2)$. Даље, $\mathbb{T}'$ поступа у зависности од следећих случајева.

1. Ако је $0 \leqslant t < g(n)$ и $X = ?$, онда се списку питања $(\star)$ додаје ново питање $(C_{i_{2t}}, C_1, g(n)-t-1, ?)$.

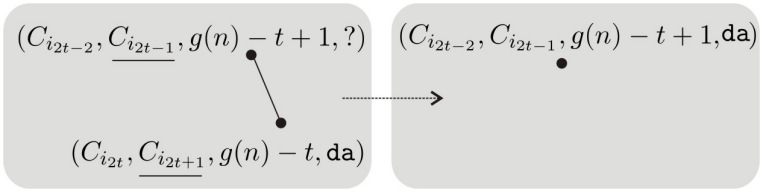
2. Ако је $t = g(n)$ и $X = ?$, онда $\mathbb{T}'$ проверава да ли је $C_{i_{2t}} = C_{i_{2t+1}}$ или се из $C_{i_{2t}}$ може добити $C_{i_{2t+1}}$ у једном кораку применом неке наредбе из $\mathbb{T}$. Уколико је одговор на једно од ова два питања потврдан, онда се питање $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, ?)$ у листи $(\star)$ замењује са $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathrm{da})$, а у супротном са $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathrm{ne})$.

3. Ако је $0 < t \leqslant g(n)$ и $X = \mathrm{da}$, разликујемо два подслучаја.

3.1. Ако је $C_{i_{2t-2}} = C_{i_{2t}}$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathrm{da})$ у $(\star)$ замењује са $(C_{i_{2t+1}}, C_{i_{2t-1}}, g(n)-t, ?)$

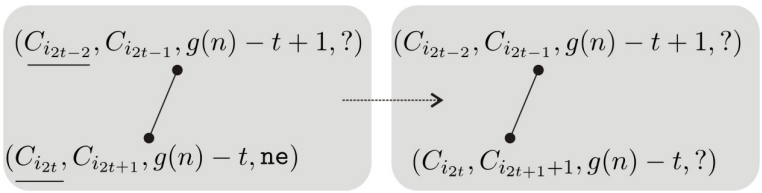


3.2. Ако је $C_{i_{2t-2}} \neq C_{i_{2t}}$, па је самим тим $C_{i_{2t-1}} = C_{i_{2t+1}}$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathtt{da})$ избацује из $(\star)$, и у претходном питању се знак ? замењује са da.

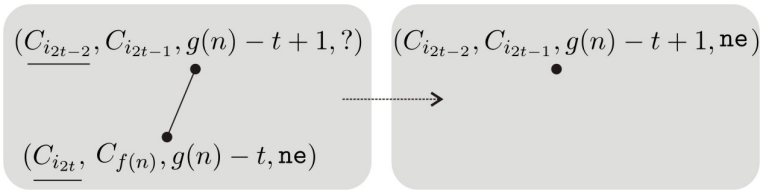


4. Ако је $0 < t \leqslant g(n)$ и $X = \mathtt{ne}$, разликујемо четири подслучаја.

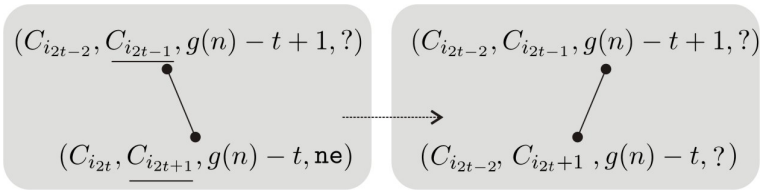
4.1. Ако је $C_{i_{2t-2}} = C_{i_{2t}}$ и $i_{2t+1} < f(n)$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathtt{ne})$ замењује у $(\star)$ питањем $(C_{i_{2t+1}}, C_{i_{2t+1}+1}, g(n)-t, ?)$, уз помоћ машине $\mathbb{E}$.



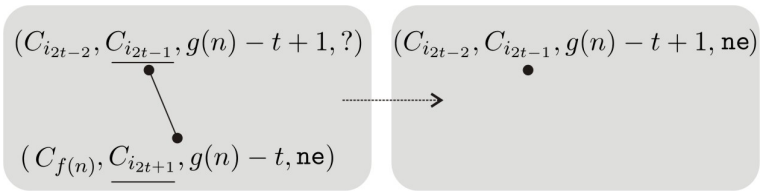
4.2. Ако је $C_{i_{2t-2}} = C_{i_{2t}}$ и $i_{2t+1} = f(n)$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathtt{ne})$ избацује из $(\star)$, и у претходном питању се знак ? замењује са ne.



4.3. Ако је $C_{i_{2t-2}} \neq C_{i_{2t}}$, па је $C_{i_{2t-1}} = C_{i_{2t+1}}$, и $i_{2t+1} < f(n)$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathtt{ne})$ замењује у $(\star)$ питањем $(C_{i_{2t-2}}, C_{i_{2t}+1}, g(n)-t, ?)$, уз помоћ машине $\mathbb{E}$.



4.4. Ако је $C_{i_{2t-2}} \neq C_{i_{2t}}$, па је $C_{i_{2t-1}} = C_{i_{2t+1}}$, и $i_{2t+1} = f(n)$, онда се $(C_{i_{2t}}, C_{i_{2t+1}}, g(n)-t, \mathtt{ne})$ избацује из $(\star)$, и у претходном питању се знак ? замењује са ne.



5. Ако је $t = 0$ и $X = \mathtt{da}$, тј. на траци је одштампано само $(C_0^w, C_{i_1}, g(n), \mathtt{da})$, разликујемо три подслучаја:

5.1. Ако је C_{i_1} конфигурација прихватања, онда $\mathbb{T}'$ враћа одговор DA;

5.2. Ако C_{i_1} није конфигурација прихватања и $i_1 < f(n)$, онда $\mathbb{T}'$ брише $(C_0^w, C_{i_1}, g(n), \mathtt{da})$, и уз помоћ машине $\mathbb{E}$, штампа питање $(C_0^w, C_{i_1+1}, g(n), ?)$;

5.3. Ако C_{i_1} није конфигурација прихватања и $i_1 = f(n)$, онда $\mathbb{T}'$ враћа одговор NE.

6. Ако је $t = 0$ и $X = \mathrm{ne}$, тј. на траци је одштампано само $(C_0^w, C_{i_1}, g(n), \mathrm{ne})$, разликујемо два подслучаја:

6.1. Ако је $i_1 < f(n)$, онда $\mathbb{T}'$ брише $(C_0^w, C_{i_1}, g(n), \mathrm{ne})$ и, уз помоћ машине $\mathbb{E}$, штампа питање $(C_0^w, C_{i_1+1}, g(n), ?)$;

6.2. Ако је $i_1 = f(n)$, онда $\mathbb{T}'$ враћа одговор NE.

Описана ТМ $\mathbb{T}'$ ради у времену $O(S(n)^2)$ и прихвата $w \in \Sigma^*$ акко $\mathbb{T}$ прихвата w. □

$$\mathbf{PSPACE} \stackrel{\text{def}}{=} \bigcup_{k \geqslant 1} \mathrm{SPACE}(n^k)$$

$$\mathbf{NPSPACE} \stackrel{\text{def}}{=} \bigcup_{k \geqslant 1} \mathrm{NSPACE}(n^k)$$

Последица 11. $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE}$

PSPACE-КОМПЛЕТНОСТ

Дефиниција 31. *Језик L је* **PSPACE***-комплетан ако:*

1. $L \in$ **PSPACE** *и*
2. *за свако* $L' \in$ **PSPACE** *важи* $L' \leq_P L$.

Сваки L који задовољава услов 2) назива се **PSPACE**-тежак.

Квантификоване исказне формуле добијамо тако што испред обичних исказних формула поставимо квантификаторе $\forall$ и $\exists$ заједно са исказним словима. Значење квантификованих исказних формула је природно. Ако је $\varphi(p, p_1, \ldots, p_n)$ исказна формула и $v : \{p_1, \ldots, p_n\} \to \{0,1\}$ валуација, онда

$$v \models \forall p\varphi(p, p_1, \ldots, p_n) \Leftrightarrow v \models \varphi(0, p_1, \ldots, p_n) \text{ и } v \models \varphi(1, p_1, \ldots, p_n)$$
$$v \models \exists p\varphi(p, p_1, \ldots, p_n) \Leftrightarrow v \models \varphi(0, p_1, \ldots, p_n) \text{ или } v \models \varphi(1, p_1, \ldots, p_n).$$

Ако су сва слова исказне формуле везана квантификаторима, онда истинитост одговарајуће квантификоване формуле не зависи од валуације. Овакве формуле називамо потпуно квантификованим формулама. Свака потпуно квантификована формула је тачна или нетачна. На пример, $\forall p \exists q((p \vee q) \wedge (\neg p \vee \neg q))$ је тачна формула. Међутим, ако заменимо места квантификаторима добијамо формулу $\exists q \forall p((p \vee q) \wedge (\neg p \vee \neg q))$ која није тачна.

Ако је $p = 0$, формула
$$(0 \vee q) \wedge (\neg 0 \vee \neg q)$$
је тачна за $q = 1$. Ако је $p = 1$, формула
$$(1 \vee q) \wedge (\neg 1 \vee \neg q)$$
је тачна за $q = 0$.

Поступајући аналогно као у Примеру 63 (страна 111), сваку потпуну квантификовану исказну формулу можемо представљати речју $\|\varphi\| \in \{0,1\}^*$.

$$\text{QBF} \stackrel{\text{def}}{=} \{\|\varphi\| \in \{0,1\}^* \mid \varphi \text{ је тачна потпуно квантификована исказна формула}\}$$

Теорема 46. QBF *је* **PSPACE***-комплетан.*

Доказ. QBF $\in$ PSPACE ...

Нека је L произвољан језик из PSPACE и $\mathbb{T} = (Q, q_0, \{q_{\text{da}}, q_{\text{ne}}\}, \Gamma, \sqcup, \Sigma, \delta)$ ТМ која одлучује L и ради у простору $p(n)$, где је p неки полином. За свако $w \in \Sigma^*$, конструисаћемо једну исказну реченицу (квантификовану исказну формулу) $\varphi_{\mathbb{T},w}$ такву да важи: $\mathbb{T}(w) \downarrow q_{\text{da}}$ акко $\varphi_{\mathbb{T},w} \in$ QBF.

Да бисмо то постигли, приметимо да за сваки улаз $w = s_1 \cdots s_n \in \Sigma^*$ дужине n, до краја израчунавања глава машине $\mathbb{T}$ не напушта регион који чине $2p(n) + 2$ суседних ћелија траке; доделимо овим ћелијама целобројне адресе $-p(n) - 1, \ldots, 0, 1, \ldots, p(n) + 1$ као у доказу Кукове теореме, при чему се улаз дужине n уноси у поља чије су адресе $1, \ldots, n$.

Постоји константа c таква да укупан број могућих текућих конфигурација није већи од $2^{cp(n)}$. Будући да се у сваком израчунавању текуће конфигурације не понављају (јер се $\mathbb{T}$ зауставља за сваки улаз), свако израчунавање машине $\mathbb{T}$ за улазе дужине n не може бити дуже од $2^{cp(n)}$.

Израчунавања машине $\mathbb{T}$ описаћемо квантификованим исказним формулама користећи слова распоређена у међусобно дисјунктне коначне скупове слова X, Y, Z_i, U_i, V_i, $1 \leqslant i \leqslant cp(n)$. Ако је I било који од наведених скупова, онда он садржи следећа исказна слова: q_a^I и s_a^I, за $q \in Q$, $s \in \Gamma$, $|a| < p(n) + 1$; словима из I можемо описати сваку појединачну конфигурацију. За било које уочене скупове I и J исказних слова уводимо следеће формуле:

- $\mathrm{START}_w(I) \equiv q_{01}^I \wedge \bigwedge_{i=1}^n s_{ii}^I \wedge \bigwedge_{a \notin \{1,\ldots,n\}} \sqcup_a^I$, при чему је $w = s_1 \cdots s_n$ улазна реч;
- $\mathrm{CONF}(I) \equiv \bigwedge_a \bigvee_s s_a^I \wedge \bigwedge_a \bigwedge_{s \neq s'} \neg(s_a^I \wedge s_a'^I) \wedge \bigvee_a \bigvee_q \left(q_a^I \wedge \bigwedge_{a' \neq a} \bigwedge_q \neg q_{a'}^I\right)$
- $\mathrm{ACC}(I) \equiv \bigvee_a q_{\mathrm{da}\,a}^I$
- $\mathrm{EQ}(I,J) \equiv \bigwedge_q \bigwedge_a \left(q_a^I \Leftrightarrow q_a^J\right) \wedge \bigwedge_s \bigwedge_a \left(s_a^I \Leftrightarrow s_a^J\right)$
- $\mathrm{NEXT}(I,J)$ је конјункција следећих формула: за све $q \in Q \setminus \{q_{\mathrm{da}}, q_{\mathrm{ne}}\}$ и $s \in \Gamma$,
  $q_a^I \wedge s_a^I \Rightarrow s_a'^J \wedge q_{a+1}'^J$ ако $\delta(q,s) = (s', \mathrm{R}, q')$, $\quad q_a^I \wedge s_a^I \Rightarrow s_a'^J \wedge q_a'^J$ ако $\delta(q,s) = (s', \mathrm{P}, q')$,
  $q_a^I \wedge s_a^I \Rightarrow s_a'^J \wedge q_{a-1}'^J$ ако $\delta(q,s) = (s', \mathrm{L}, q')$, $\quad s_a^I \wedge \bigwedge_q \neg q_a^I \Rightarrow s_a^J$.

Рекурзивно дефинишемо низ квантификованих исказних формула $\mathrm{REACH}_i(I,J)$, $0 \leqslant i \leqslant cp(n)$, чије је значење: из конфигурације која је описана словима из I може се стићи у конфигурацију описану словима из J у највише 2^i корака израчунавања. Нека је:

$$\mathrm{REACH}_0(I,J) \equiv \mathrm{CONF}(I) \wedge \mathrm{CONF}(J) \wedge (\mathrm{EQ}(I,J) \vee \mathrm{NEXT}(I,J)).$$

Усвајамо следеће поједностављење нотације: ако је $I = \{p_1, \ldots, p_n\}$ коначан скуп исказних слова, писаћемо $\exists I$ уместо $\exists p_1 \cdots \exists p_n$ и $\forall I$ уместо $\forall p_1 \cdots \forall p_n$. Формула $\mathrm{REACH}_{i+1}(I,J)$ би се могла дефинисати као $\exists Z_i(\mathrm{CONF}(Z_i) \wedge \mathrm{REACH}_i(I,Z_i) \wedge \mathrm{REACH}_i(Z_i,J))$, међутим тада ће $\mathrm{REACH}_{cp(n)}(I,J)$ бити експоненцијалне дужине, па је не можемо записати у полиномијалном времену. Овај проблем избегавамо коришћењем следеће еквивалентне формуле:

$$\begin{aligned}\mathrm{REACH}_{i+1}(I,J) \equiv \exists Z_i(\mathrm{CONF}(Z_i) \wedge \forall U_i \forall V_i((\mathrm{EQ}(U_i,I) \wedge \mathrm{EQ}(V_i,Z_i)) \vee \\ \vee (\mathrm{EQ}(U_i,Z_i) \wedge \mathrm{EQ}(V_i,J)) \Rightarrow \mathrm{REACH}_i(U_i,V_i)).\end{aligned}$$

Жељена формула је $\varphi_{\mathbb{T},w} \equiv \exists X \exists Y(\mathrm{START}_w(X) \wedge \mathrm{REACH}_{cp(n)}(X,Y) \wedge \mathrm{ACC}(Y))$. □

Игре су веома блиско повезане са квантификаторима уопште. Посматрајмо следећу игру. Нека је

$$\varphi = Q_1 p_1 Q_2 p_2 Q_3 p_3 \ldots Q_k p_k \theta(p_1, p_2, p_3, \ldots, p_k),$$

потпуно квантификована исказна формула, $Q_i \in \{\forall, \exists\}$. Формули φ придружујемо следећу игру између играча A и играча E: у i-том кораку, ако је $Q_i = \forall$, игра играч A тако што додељује произвољну вредност из $\{0,1\}$ слову p_i; а ако је $Q_i = \exists$, играч E додељује произвољну вредност из $\{0,1\}$ слову p_i. Игра се завршава после k корака и

- побеђује играч E ако је формула θ ТАЧНА за изабране вредности исказних слова;
- побеђује играч A ако је формула θ ЛАЖНА за изабране вредности исказних слова.

На пример, нека је γ_1 формула

$$\exists p_1 \forall p_2 \exists p_3((p_1 \vee p_2) \wedge (p_2 \vee p_3) \wedge (\neg p_2 \vee \neg p_3)).$$

Тада E има победничку стратегију: прво узима $p_1 = 1$, а затим узима вредност p_3 као негацију онога што је изабрао A за p_2. За формулу γ_2:

$$\exists p_1 \forall p_2 \exists p_3((p_1 \vee p_2) \wedge (p_2 \vee p_3) \wedge (p_2 \vee \neg p_3)),$$

играч A има победничку стратегију: без обзира шта E бира, A узима $p_2 = 0$.

GAME $\stackrel{\text{def}}{=} \{[\![\varphi]\!] \mid E$ има победничку стратегију у игри одређеној формулом $\varphi\}$

Теорема 47. GAME *је* **PSPACE**-*комплетан.*

Игра GO**.** Нека је G произвољан усмерен граф и s један чвор. Играчи I и II играју следећу игру:

- игру започиње играч I са стартне позиције s;
- играчи наизменично бирају једног непосредног суседа текућег чвора;
- током игре се чворови не смеју понављати;
- губи онај играч који не може да настави игру (и наравно, побеђује онај други).

GO $\stackrel{\text{def}}{=} \{[\![G, s]\!] \mid$ Играч I има победничку стратегију$\}$

Теорема 48. GO *је* **PSPACE**-*комплетан.*

Скица доказа. GO $\in$ PSPACE ...

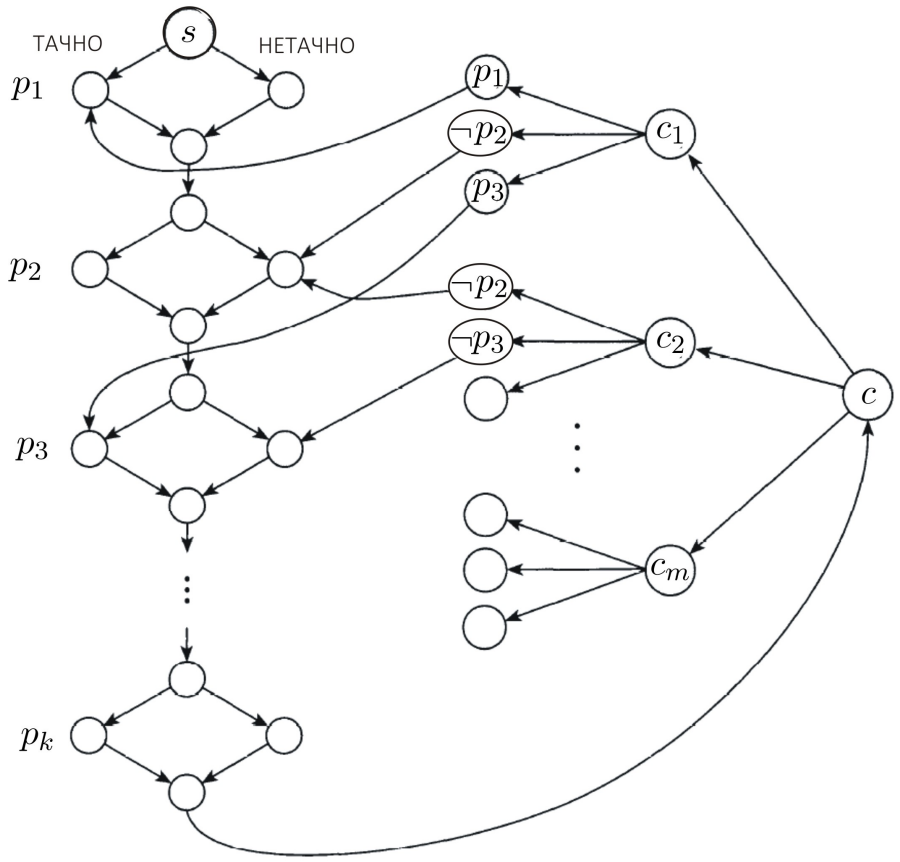
Довољно је доказати да је GAME $\leq_{\text{P}}$ GO. За произвољну потпуно квантификовану формулу φ треба конструисати усмерен граф G са истакнутим почетним чвором s тако да важи:

Играч E има победничку стратегију у φ-игри $\Leftrightarrow$ Играч I има победничку стратегију у (G, s)-игри .

Без губљења општости се може претпоставићемо да је φ облика:

$$\exists p_1 \forall p_2 \exists p_3 \forall p_4 \cdots \exists p_k \theta,$$

при чему је θ у КНФ. Увођењем „лажних“ слова, какав год да је префикс формуле φ, она се може прерадити у наведени облик (са наизменичним смењивањем квантификатора $\exists$ и $\forall$, при чему је $\exists$ први и последњи квантификатор). Општи поступак прераде формула оваквог облика у жељени граф са стартним чвором илустрован је наредном сликом.



$$\exists p_1 \forall p_2 \exists p_3 \forall p_4 \cdots \exists x_k \left(\underbrace{(p_1 \vee \neg p_2 \vee p_3)}_{c_1} \wedge \underbrace{(\neg p_2 \vee \neg p_3 \vee \cdots)}_{c_2} \wedge \cdots \wedge \underbrace{(\quad)}_{c_m} \right)$$