1 Cvičenie 3

Keď už máme zavedené čo to znamená niečo počítať (while programy) a predstavu o tom čo všetko sa dá a nedá spočítať (vyčísliteľné a nevyčísliteľné funkcie), môžeme sa zaoberať ďalšou dôležitou otázkou: Čo je to riešiteľný problém, prípadne či existujú rôzne druhy "riešiteľnosti" ktoré by nás mohli zaujímať.

Na to si najskôr musíme samozrejme pripomenúť čo to vôbec je problém. Obecne budeme predpokladať že problém A je ľubovoľná podmnožina prirodzených čísel $A \subseteq \mathbb{N}$. Teda problém môže byť napr. množina $A_1 = \{x \mid x \text{ je deliteľné } 13\}$ alebo $A_2 = \{x \mid \varphi_x(1) = 42\}$ (indexy funkcií ktoré pre vstup 1 vracajú 42). Samozrejme, v informatike často pracujeme aj s inými typmi objektov ako sú prirodzené čísla, napr. grafy alebo logické formule. Takéto objekty sa ale dajú kódovať pomocou prirodzených čísel (graf môže byť reprezentovaný ako matica susednosti, zoznam hrán a vrcholov, atď.) a teda sa môžeme často stretnúť aj s takto definovanými problémami: $A_3 = \{ \text{ graf } G \mid G \text{ obsahuje cyklus dĺžky } 4\}$ alebo $A_4 = \{ \text{ výroková formula } \psi \mid \psi \text{ je tautológia } \}$.

1.1 Rozhodnuteľné problémy

Prirodzená otázka je potom čo to znamená "rozhodnúť problém?". Za týmto účelom definujeme triedu rozhodnuteľných (rekurzívnych) problémov (množín): Množina A je rekurzívna práve vtedy keď existuje totálne vyčísliteľná funkcia f_A t.ž. $f_A(x) = 1 \Leftrightarrow x \in A$. Teda f_A pre každý vstup vždy skončí a vracia jednotku len ak vstup patrí do A. Teda podľa výsledku funkcie f_A vieme presne určiť či prvok x patrí alebo nepatrí do množiny A. (Existuje aj ekvivalentná definícia ktorá ešte navyše tvrdí že f_A vracia 0 ak x do A nepatrí)

Ako ukázať že množina A je rekurzívna? (problém je rozhodnuteľný) Najjednoduchší spôsob je samozrejme nájsť totálnu funkciu f_A ktorá náš problém rozhoduje. Pokiaľ toto riešime pre konkrétny problém, znamená to vyriešiť daný problém (opäť nás nezaujíma zložitosť daného riešenia, iba že je totálne). Napr. množina A_1 je rekurzívna, keďže stačí zistiť zbytok po delení x číslom 13 a otestovať či je výsledok nula.

Príklad: Nech B_1 a B_2 sú rekurzívne množiny. Dokážte že $B_U = B_1 \cup B_2$, $B_I = B_1 \cap B_2$ a $B_C = \overline{B_1}$ (doplnok) sú rekurzívne množiny. Tu nám z predpokladu rekurzívnosti plynie existencia rozhodovacích funkcií f_{B_1} a f_{B_2}. Pomocou týchto funkcií potom potrebujeme vytvoriť rozhodovacie funkcie pre B_U, B_I a B_C. Uvažujme $f_{B_U}(x) = \begin{cases} 1 & f_{B_1}(x) = 1 \vee f_{B_2}(x) = 1 \\ 0 & \text{inak} \end{cases}$, $f_{B_I}(x) = \begin{cases} 1 & f_{B_1}(x) = 1 \wedge f_{B_2}(x) = 1 \\ 0 & \text{inak} \end{cases}$ a $f_{B_C}(x) = \begin{cases} 1 & f_{B_1}(x) \neq 1 \\ 0 & \text{inak} \end{cases}$. Tieto tri funkcie sú zjavne vyčísliteľné (ako jednoduché cvičenie si ich môžete skúsiť naprogramovať vo while programoch) a sú taktiež totálne (za predpokladu že f_{B_1} a f_{B_2} sú totálne a vyčísliteľné, čo vieme že sú). Ich korektnosť potom pomerne triviálne plynie z definície prieniku, zjednotenia a doplnku. Ako ukážku dokážeme korektnosť f_{B_U}:

$[f_{B_U}(x) = 1] \Leftrightarrow [f_{B_1}(x) = 1 \vee f_{B_2}(x) = 1] \Leftrightarrow [x \in B_1 \vee x \in B_2] \Leftrightarrow [x \in B_1 \cup B_2] \Leftrightarrow [x \in B_U]$ – Prvá ekvivalencia plynie z definície f_{B_U}, druhá z definície rekurzívnej množiny, tretia je definícia zjednotenia a štvrtá je len definícia B_U.

Ako ukázať že množina nie je rekurzívna? Rekurzívne množiny by zrejme nemali veľký význam keby všetky množiny boli rekurzívne. Aby sme našli nejaké množiny ktoré nie sú rekurzívne, môžeme sa zamerať na iné objekty o ktorých "neexistencií" už vieme - nevyčísliteľné funkcie:

Uvažujme množinu $K = \{x \mid \varphi_x(x) \neq \perp\}$ (φ_x zastaví pre vstup x). Množinu K budeme nazývať problém zastavenia. Množina K nie je rekurzívna. Dôkaz: Pre spor predpokladajme že K je rekurzívna, teda existuje funkcia f_K ktorá ju rozhoduje. Potom si ale môžeme všimnúť že $halt = f_K$, teda ak f_K je vyčísliteľná, je vyčísliteľná aj funkcia *halt* o ktorej sme na minulom cvičení dokázali opak, čo je spor. K teda nemôže byť rekurzívna.

Keď už teda vieme že existuje nerekurzívna množina K, môžeme jej existenciu použiť pri argumentácií o iných podobných množinách:

Príklad: Dokážte že množina $A_2 = \{x \mid \varphi_x(1) = 42\}$ nie je rekurzívna: Pre spor predpokladajme že A_2 rekurzívna je, a teda existuje jej rozhodovacia funkcia f_{A_2}. Uvažujme ďalej funkciu $g(x,y) = \begin{cases} 42 & \varphi_y(y) \neq \perp \\ \perp & \text{inak} \end{cases}$ (teda jej výstup vôbec nezáleží na x). Táto funkcia je vyčísliteľná (spustíme $\varphi_y(y)$ pomocou univerzálnej funkcie, ak skončí, vrátime 42) a teda má nejaký index, napr. e (teda $g = \varphi_e$). Podľa vety o parametrizácií existuje funkcia s pre ktorú platí $\varphi_{s(i,y)}(x) = \varphi_i(x,y)$. Pomocou týchto pozorovaní môžeme zostaviť rozhodovaciu funkciu pre problém zastavenia: Funkcia f_K pre vstup y spočíta hodnotu $a = s(e,y)$ a následne vráti $f_{A_2}(a)$ (teda zistí či a patrí do A_2 alebo nie). Ak $\varphi_y(y)$ zastaví, tak funkcia daná indexom a zastaví a vráti 42, teda patrí do A_2. Ak $\varphi_y(y)$ nezastaví, tak funkcia daná indexom a tiež nezastaví, teda nepatrí do A_2. Celkovo teda dostávame že $f_K(y) = 1 \Leftrightarrow \varphi_y(y) \neq \perp \Leftrightarrow y \in K$. To je ale spor s práve dokázaným tvrdením že K nie je rekurzívna množina, teda náš predpoklad o rekurzívnosti A_2 bol nesprávny.

Samozrejme, dokazovať takýmto spôsobom ne-rekurzívnosť je pomerne zdĺhavé a komplikované. Uľahčiť si to môžeme pomocou prvej Riceovej vety. Neformálne si môžeme všimnúť, že rovnosť dvoch funkcií nie je vyčísliteľná, teda ak by sme ju pri vyhodnocovaní nášho problému museli vyriešiť, asi sa nebude jednať o rekruzívnu množinu (toto je samozrejme ale len intuitívny argument, presný dôkaz by vyzeral inak). Formálne sa táto vlastnosť nazýva zachovávanie funkcií. Hovoríme že množina zachováva funkcie ak pre každú funkciu platí že množina buď obsahuje všetky jej indexy, alebo žiadny (každá vyčísliteľná funkcia má nekonečne veľa indexov).

Napríklad množina A_2 funkcie zachováva (funkcia pre vstup 1 vracia 42 bez ohľadu na to aký index tejto funkcie uvažujeme). Na druhej strane, množina K funkcie nezachováva. Uvažujme funkciu ktorá cyklí pre všetky možné vstupy okrem jednej konštanty c pričom táto konštanta je zároveň aj jedným z indexov jej programov. Potom $c \in K$, ale všetky ostatné indexy tejto funkcie do K nepatria.

Prvá Riceova veta potom hovorí, že ak je množina A netriviálna (nie je prázdna ani $\mathbb{N}$) a zachováva funkcie, nie je rekurzívna. A_2 je zjavne netriviálna (neobsahuje napr. index prázdnej funkcie, teda nie je $\mathbb{N}$, ale obsahuje indexy funkcií ktoré konštantne vracajú 42, teda nie je prázdna) a zároveň ako bolo spomenuté vyššie, zachováva funkcie (formálny dôkaz sa dá viesť pomerne jednoducho: Nech a a b sú dva rôzne indexy tej istej funkcie. Potom $\varphi_a(1) = \varphi_b(1)$ a táto hodnota buď je 42 alebo nie, teda ak a patrí do A_2 tak aj b patrí do A_2 a opačne). Teda podľa prvej Riceovej vety nie je množina A_2 rekurzívna.

Ako sme videli na príklade množiny K, nie všetky ne-rekurzívne množiny zachovávajú funkcie, teda prvá Riceova veta sa nedá použiť vždy (v takých prípadoch potom treba siahnuť po podobnom prístupe ako sme pre A_2 použili pôvodne).

1.2 Čiastočne rozhodnuteľné problémy

Logickou otázkou čo by sme si mohli ďalej položiť je: Existuje niečo medzi rozhodnutenými a nerozhodnuteľnými problémami? Napr. problém zastavenia síce nie je rekurzívny, na druhej strane ale pre každý program ktorý zastavuje vieme potvrdiť že skutočne zastaví. Problém je v tom že nevieme potvrdiť aj nezastavenie. Teda ak je odpoveď pozitívna, vieme ju dokázať.

Táto myšlienka je potom formalizovaná v koncepte čiastočne rozhodnutených problémov (res. rekurzívne spočetných množín, často sa používa skratka R.E. – recursively enumerable). R.E. množiny sa dajú definovať niekoľkými spôsobmi, pričom tieto definície sú ekvivalentné, len v niektorých prípadoch môže byť niektorá z nich praktickejšia:

1. Množnia A je rekurzívne spočetná ak existuje vyčísliteľná funkcia f t.ž. $A = domain(f)$. Teda A je definičným oborom nejakej vyčísliteľnej funkcie.

2. Množina A je rekurzívne spočetná ak je prázdna alebo existuje (totálne) vyčísliteľná funckia f t.ž. $A = range(f)$. Teda A je oborom hodnôť nejakej (totálne) vyčísliteľnej funkcie, alebo, inak povedané, existuje funkcia f ktorá je numeráciou množiny A. (f sa často uvažuje ako totálna, ale v skutočnosti to nie je nutné - z netotálnej numerácie sme potom schopní vyrobiť pomocou malého triku aj totálnu)

Jazykové okienko: Prečo rekurzívne a rekurzívne spočetné množiny? Jedným z možných výpočetných modelov okrem while programov sú aj tzv. primitívne rekurzívne funkcie. No a rekurzívna množina je teda rekurzívna preto, že existuje rekurzívna funkcia ktorá rozhoduje príslušnosť do tejto množiny. Rekurzívne spočetná množina (v angličtine recursively enumerable) je potom množina ktorá je numerovateľná rekurzívnou funkciou, preto enumerable, aj keď teda ten preklad nie je najšťastnejší.

Každá rekurzívna množina je aj R.E. Keďže pre rekurzívnu množinu máme jasne spočítateľné kedy do nej prvok patrí a kedy nie, nie je problém skonštruovať funkciu ktorá sa zacyklí keď prvok do množiny nepatrí, a tým splníme podmienky prvej definície. Niekedy sa teda môžeme v zadaní stretnúť s pojmami ako "množina je R.E. ale nie je rekurzívna" aby bolo jasné že pre túto množinu nie len že nemusí ale ani nemôže existovať exaktná rozhodovacia funkcia. Na druhej strane, keď sa povie že množina je R.E. bez nejakých prívlastkov, neznamená to automaticky že nemôže byť aj rekurzívna.

Ako ukázať že množina je R.E.? Podobne ako pri rekurzívnych množinách, ak máme daný konkrétny problém, najjednoduchšie je nájsť funkciu ktorá by spĺňala jednu z našich definícií. Napr. pre problém zastavenia je to triviálne: Pokiaľ naša f pre vstup x spustí pomocou univerzálnej funkcie $\varphi_x(x)$, bude platiť že $domain(f) = K$. Pokiaľ riešime nejaké obecnejšie tvrdenie, musíme zase vychádzať z definície, podobne ako pri rekurzívnych množinách (akurát si môžeme vybrať ktorú definíciu použijeme – prípadne ich môžeme aj striedať).

Príklad: Nech B_1 a B_2 sú R.E. množiny. Dokážte že $B_U = B_1 \cup B_2$ a $B_I = B_1 \cap B_2$ sú R.E. množiny.

Pre množinu B_U skonštruujeme d_{B_U} nasledovne: Nech r_{B_1} a r_{B_2} sú funkcie podľa definície (2). Potom d_{B_U} pre vstup x spočíta či je x sudé alebo liché, ak je sudé, spustí a vráti $r_{B_1}(\frac{x}{2})$, ak je liché, spustí a vráti $r_{B_2}(\frac{x-1}{2})$. Teda ak y patrí do oboru hodnôt r_{B_1} alebo r_{B_2}, bude patriť aj do oboru hodnôť r_{B_U} (pre vstup $2y$, res. $2y+1$ podľa toho či patrí do prvej alebo druhej množiny) a teda sĺňame definíciu (2). (Všetko za predpokladu že množiny sú neprázdne - ak je niektorá prázdna, výsledkom je funkcia neprázdnej množiny, ak sú obe prázdne, výsledok je R.E. z definície)

Pre množinu B_I skonštruujeme d_{B_I} nasledovne: Nech d_{B_1} a d_{B_2} sú funkcie podľa definície (1), potom funkcia d_{B_I} pre vstup x najskôr spustí $d_{B_1}(x)$ a potom $d_{B_2}(x)$ – návratové hodnoty sú irelevantné, ide nám len o to či sa zacyklia alebo nie. Pokiaľ x patrí do definičných oborov oboch funkcií (teda patrí do oboch množín), tak funkcia skončí, inak sa zacyklí, teda spĺňame definíciu (1).

R.E. množiny nie sú uzavreté na doplnok. Sporom: Uvažujme netriviálnu R.E. množinu (neprázdna a nie je $\mathbb{N}$) B a jej doplnok $\overline{B}$. Pre spor predpokladajme že $\overline{B}$ je tiež R.E. množina. Potom existujú numerácie r_B a $r_{\overline{B}}$ podľa definície (2) – pre jednoduchosť si ich nazveme r_1 a r_2. Uvažujme nasledujúci program:

```
input: x; output: y;
begin
        found := 0; n := 0;
        while found = 0 do
        begin
                if r1(n) = x || r2(n) = x do
                        found := 1;
                n := n + 1;
        end
        if r1(n-1) = x do y := 1; else y := 0;
end
```

Tento program potom definuje rozhodovaciu funkciu pre množinu B. Prečo? O každom čísle x vieme že patrí buď do B alebo $\overline{B}$. Teda sa nutne musí objaviť v obore hodnôt r_1 alebo r_2 (a samozrejme sa nemôže objaviť v oboch). Teda čo môžeme robiť je postupne prechádzať obe numerácie a hľadať v ktorej z nich sa nachádza x. Ak je to v numerácií B, vraciame 1, ak je to numerácia $\overline{B}$, vraciame 0.

Teda sme ukázali že ak by boli všetky R.E. množiny uzavreté na doplnok, tak sú všetky rekurzívne. To by ale znamenalo že aj problém zastavenia je rekurzívny, čo je spor. Zaujímavý dôsledok tejto vlastnosti je aj to, že sme práve ukázali existenicu množín ktoré nie sú ani rekurzívne - sú to (okrem iného) doplnky nerekurzívnych R.E. množín. Teda doplnok problému zastavenia, $\overline{K}$, nie je ani R.E. (dáva to zmysel aj intuitívne, keďže to sú programy ktoré nezastavujú nad svojím vlastným indexom - teda nemám spôsob ako potvrdiť že program nezastaví).

Step counter: Počas dokazovania rekurzívnej spočetnosti budeme často potrebovať simulovať výpočet programu pre konečný počet krokov (teda intuitívne "spustiť program s timeoutom"). Tento simulátor sa nezýva step counter Sc a ide v podstate o upravenú univerzálnu funkciu (teda to že ide o totálne vyčísliteľnú funkciu asi zase nikoho veľmi neprekvapí):

$$Sc(i,x,k) = \begin{cases} 1 & \varphi_i(x) \text{ zastaví do } k \text{ krokov} \\ 0 & \text{inak} \end{cases}$$

Uvažujme množinu A_2. Pomocou definície (1) je opäť jednoduché ukázať že množina je R.E. – stačí spustiť funkciu na vstupe pre hodnotu 1 a počkať či vráti 42, ak nie cykliť. Čo ak by sme ale túto množinu trochu upravili: $A_3 = \{i \mid 42 \in range(\varphi_i)\}$ – nemáme pevne daný vstup pre ktorý sa má vrátiť 42. Triviálne by sme mohli spúšťať funkciu i pre rôzne vstupy a hľadať 42, pokiaľ by sme ale narazili na vstup kde i cyklí, nemáme sa ako posunúť ďalej.

Teda musíme pokusy vo vhodnom čase zastaviť a skúšať ďalšie hodnoty:

```
input i;
begin
    n := 0; found := 0;
    while found = 0 do
    begin
        x := n; k := 0;
        while k != n + 1 do:
        begin
            if Sc(i, x, k) = 1 and phi_i(x) = 42 do
                found := 1;
            k := k + 1; x := x - 1;
        end
        n := n + 1;
    end
end
```

(Predpokladáme že logické spojky v if sa vyhodnocujú lenivo - teda $\varphi_i(x)$ sa spočíta až keď step counter dopredu potvrdí že výpočet s týmto vstupom skončí)

Čo sa to tu vlastne deje? Vieme že ak i patrí do A_3, existuje vstup x a počet krokov k, taký že $\varphi_i(x)$ zastaví do k krokov a vráti 42. Na to aby sme ho našli, potrebujeme vyskúšať všetky dvojice hodnôt x a k. Teda potrebujeme postupne prechádzať tebuľku dvojíc hodnôt. Pokiaľ by sme niečo také skúšali robiť po stĺpcoch/riadkoch, nepodarí sa nám to, lebo každý riadok/stĺpec je nekonečný. Čo ale môžeme spraviť je prechádzať ju po diagonálach. Teda najskôr začneme kombináciou $(0,0)$. Následne sa posunieme na druhú diagonálu $(n=1)$ a vyskúšame dvojice $(0,1)$ a $(1,0)$. Na tretej diagonále $(n=2)$ skúšame $(0,2),(1,1)$ a $(2,0)$. Ďalej to budú $(0,3),(1,2),(2,1)$ a $(3,0)$.

Takýmto spôsobom vieme, že nech už je kombinácia k a x ktorá nám vráti 42 ľubovoľná, nachádza sa na nejakej diagonále, my sa k nej v konečnom čase prepracujeme (lebo každá diagonála, narozdiel od riadku/stĺpca je konečná) a nájdeme ju. Ak taká dvojica neexistuje, budeme samozrejme tabuľku prechádzať stále ďalej a ďalej bezúspešne a efektívne sa nám program zacyklí. Teda sme zostrojili funkciu ktorej definičný obor je A_3.

Takáto operácia je pomerne bežná pri práci s R.E. množinami, teda často predpokladáme existenciu tzv. projekčných funkcií π_1 a π_2 ktoré nám dokážu unikátne "z jednoho čísla vyrobiť dve" (inak povedané, máme bijekciu medzi číslami a dvojicami čísel). Máme nejaké číslo ktoré nám identifikuje pozíciu dvojice v tabuľke (počítanú podobným spôsobom ako v predchádzajúcom príklade), a π_1 a π_2 nám vrátia prvú a druhú hodnotu tejto dvojice. Teda v našom príklade $id(0,0) = 0$, $id(0,1) = 1$, $id(1,0) = 2$, $id(0,2) = 3$, $id(1,1) = 4$, $id(2,0) = 5$, atď. a teda projekcia nám vráti napr $\pi_1(4) = 1$, $\pi_2(4) = 1$, $\pi_1(3) = 0$, $\pi_2(3) = 2$, atď.

Pomocou step counteru a projekcie môžeme potom napr. ukázať ako z netotálnej numerácie v definícií (2) urobiť totálnu (a teda že definície bez a s totálnosťou sú ekvivalentné). Nech A je neprázdna množina, r je index jej netotálnej numerácie a a je nejaký prvok z tejto množiny. Uvažujme potom napr. takýto program:

```
input x; output y;
begin
    if Sc(r, pi1(x), pi2(x)) = 1 do
        y := phi_r(pi1(x));
    else
        y := a;
end
```

Takáto numerácia vždy zastaví, no a ak pôvodná numerácia vracala nejaký prvok y pre vstup x po k krokoch, tak táto numerácia ho vráti pre vstup $id(x,k)$. Teda obor hodnôt tejto numerácie je zhodný s pôvodným oborom hodnôt.

Ako ukázať že množina nie je R.E.? Tu má človek hneď niekoľko možností, podobne ako v predchádzajúcom prípade:

Môžeme ukázať že doplnok tejto množiny je R.E. ale nie je rekurzívny. Z neuzavretosti na doplnok potom máme že naša množina nemôže byť R.E. ak už jej doplnok je R.E. Toto ale vždy nemusí fungovať, keďže existujú aj množiny kde ani jednen z dvojice množina-doplnok nie je R.E. Napríklad množina $A_4 = \{x \mid \varphi_x \text{ je totálna}\}$ intuitívne asi nebude R.E. (obecne nemám ako potvrdiť že funkcia je totálna), ale jej doplnok $\overline{A}_4$ tiež asi nebude R.E. (totálnosť nemám totiž ani ako vyvrátiť, musel by som rozhodnúť kedy funkcia nezastavuje).

Druhá možnosť je opäť použiť existujúce ne-R.E. množiny pri dôkaze, podobne ako pri nerekurzívnosti A_2: Pre spor predpokladajme že $\overline{A_4}$, teda indexy všetkých netotálnych funkcií, je R.E., a teda máme funkciu $d_{\overline{A_4}}$ t.ž. $A_4 = domain(d_{\overline{A_4}})$. Potom ale uvažujme funkciu $d_{\overline{K}}$, ktorá pre vstup x vyprodukuje index a programu ktorý bez ohľadu na vstup počíta vždy funkciu $\varphi_x(x)$ (podobne ako v dôkaze pre A_2 môžeme použiť vetu o parametrizácií). S týmto indexom na vstupe potom spustí funkciu $d_{\overline{A_4}}$. Ak $d_{\overline{A_4}}$ zastaví, znamená to že a počíta funkciu ktorá nie je totálna a teda že $\varphi_x(x)$ cyklí. Ak nezastaví, znamená to že a počíta totálnu funkciu a teda $\varphi_x(x)$ zastaví. Teda $domain(d_{\overline{K}}) = \overline{K}$, čo je spor s tým že $\overline{K}$ nie je R.E., teda náš predpoklad o rekurzívnosti $\overline{A_4}$ bol nesprávny.

Podobne ako v prípade rekurzívnych množín si tento proces môžeme zjednodušiť pomocou Riceovej vety, avšak v tomto prípade budeme mať dve varianty:

- Druhá Riecova veta: Ak A je netriviálna a rešpektuje funkcie (teda nie je rekurzívna podľa prvej Riceovej vety) a zároveň platí že existujú dve funkcie φ_i a φ_j takže že $i \in A$ a $j \notin A$ pričom φ_j je rozšírením φ_i, tak A nie je R.E. (funkcia g rozširuje funkciu f ak je g definovaná všade rovnako ako f, ale môže byť definovaná aj pre hodnoty kde f definovaná nie je).

  Príklad: Funkcia *halt* je rozšírením funkcie $f(x) = \begin{cases} 1 & \varphi_x(x) \neq \perp \\ \perp & \text{inak} \end{cases}$ (halt je definovaná aj keď $\varphi_x(x)$ nezastaví). Akákoľvek funkcia je rozšírením prázdnej funkcie (môžem pridať úplne hocičo). Jediným rozšírením totálnej funkcie je táto funkcia sama (nemám čo viac pridať).

  Čo to vlastne znamená? Znamená to že ak existuje netotálna funkcia ktorá patrí do A, ale rozšírením tejto netotálnej funkcie sa viem dosať "von" z A, tak A nie je R.E. To okrem iného znamená že táto Riceova veta sa nedá použiť na dôkaz množiny A_4. Keďže všetky funkcie v A_4 sú totálne, nemám ich ako rozšíriť.

  Dá sa ale použiť na dôkaz množiny $\overline{A_4}$. Uvažujme prázdnu funkciu a jej rozšírenie - identitu. Prázdna funkcia nie je totálna, teda patrí do $\overline{A_4}$, ale identita je totálna, teda do $\overline{A_4}$ nepatrí. $\overline{A_4}$ teda nemôže byť R.E. (samozrejme ešte treba overiť že $\overline{A_4}$ je netriviálna a rešpektuje funkcie, to je ale ľahké, keďže sme práve našli funkciu čo tam patrí a funkciu čo tam nepatrí, no a ak je funkcia netotálna, všetky jej indexy sú netotálne).

  Hint: Pri hľadaní vhodnej funkcie je dobré začať s "najmenšou" (najmenej definovanou) funkciou ktorá patrí do A. To je často krát prázdna funkcia (ak prázdna funkcia patrí do netriviálnej množiny a množina zachováva funkcie, tak nemôže byť R.E. – ľubovoľná funkcia ktorá nie je v A bude rozšírením prázdnej funkcie a teda splňovať podmienky). Ak do množiny prázdna funkcia nepatrí, dá sa "najemnšia" funkcia často odvodiť z definície. Tiež sa dá zamerať na konečné vs. nekonečné funkcie.

- Tretia Riceova vera: Ak A je netriviálna a rešpektuje funkcie (teda nie je rekurzívna) a zároveň existuje funkcia φ_i t.ž. $i \in A$ a pre každé konečné zúženie φ_j platí $j \notin A$, tak A nie je R.E. (zúženie je opačný pojem k rozšíreniu – odoberám prvky miesto pridávania; konečné zúženie je zúženie ktorého definičný obor je konečný).

  Príklad: Funkcia ktorá rozhoduje problém zastavenia pre prvých 1000 programov je konečným zážením funkcie *halt* (a je dokonca vyčísliteľná, lebo všetky funkcie s konečným definičným oborom sú vyčísliteľné). Prázdna funkcia má jediné zúženie: samú seba.

  Čo to teda znamená? Ak máme nejakú funkciu ktorá patrí do A, ale všetky jej "konečné podčasti" do A nepatria, tak A nie je R.E. Taktiež si všimnime že ak množina obsahuje prázdnu funkciu, nedá sa táto veta použiť, lebo prázdna funkcia je konečným zúžením ľubovoľnej funkcie.

  Dá sa ale použiť na dôkaz množiny A_4. Identita je totálna funkcia ktorá patrí do A_4, ale jej ľubovoľné konečné zúženie nie je totálne, teda do A_4 nepatrí (A_4 dokonca neobsahuje žiadnu konečnú funkciu, takže je úplne jedno akú funkciu z A_4 si vyberieme, žiadne konečné zúženie tam nikdy nebude). Netriviálnosť a zachovávanie funkcií je jasné, A_4 teda nie je R.E.

  Hint: Pri hľadaní vhodnej funkcie je dobré zase hľadať čo najjednoduchšiu neprázdnu funkciu ktorá patrí do A. Častokrát si ale človek vystačí s identitou alebo rôznymi konštantnými funkciami. Dôležité ale je, že funkcia musí mať nekonečný definičný obor, inak je sama sebe konečným zúžením a teda nikdy nemôže fungovať.